rendu_de_monnaie

March 29, 2020

notebook consultable, exécutable, modifiable et téléchargeable en ligne :

- se rendre à : https://github.com/nsi-acot/continuite_pedagogique_premiere
- cliquer sur l'icone "launch binder" en bas de page
- patienter quelques secondes que le serveur Jupyter démarre
- naviguer dans le dossier `"./algorithmique/algos_gloutons/"`
- cliquer sur le nom de ce notebook

1 Rendu de monnaie

1.0.1 Systeme de monnaie

Un achat en espèces se traduit par un échange de pièces et de billets. Dans la suite, **ce qu'on appellera des *pièces* désignera aussi bien les véritables pièces que les billets**.

Dans le système monétaire de la zone euro, si on se limite aux sommes entières (pas de centimes) les pièces prennent pour valeurs 1, 2, 5, 10, 20, 50, 100, 200 et 500 euros . On dit que le système de monnaie peut être représenté par le tableau

```
systeme_monnaie_euro = [1, 2, 5, 10, 20, 50, 100, 200, 500]
```

Néanmoins on pourrait considérer d'autres ensembles de monnaie. Par exemple le tableau

```
systeme_monnaie_pluton = [1, 3, 6, 12, 24, 30]
```

1.0.2 Exemple avec une somme à rendre de 49

Supposons maintenant qu'on doive rendre 49 euros de monnaie. Quelles pièces peuvent être rendues ? La réponse n'est pas unique.

- Avec `systeme_monnaie_euro` : deux pièces de 20, 1 pièce de 5 et deux pièces de 2 conviennent. Mais quarante-neuf pièces de 1 conviennent aussi.
- Avec `systeme_monnaie_pluton` : une pièce de 30, une pièce de 12, une pièce de 6 et une pièce de 1 conviennent. Mais une pièce de 10 et treize pièces de 3 conviennent également.

Remarque : Dans tout ce notebook, on suppose que pour rendre la monnaie on dispose d'une "caisse" contenant un nombre infini de chacune des pièces du système de monnaie choisi.

1.0.3 Minimiser le nombre de pièces à rendre

Si on souhaite maintenant rendre la monnaie `somme_a_rendre = 49` **avec un minimum de pièces**, on peut démontrer que : - pour `systeme_monnaie_euro`, la meilleure solution pour rendre

49 est `liste_rendu = [20, 20, 5, 2, 2]`.

- pour `systeme_monnaie_pluton`, la meilleure solution est `liste_rendu = [24, 24, 1]`.

1.0.4 Définition du problème du rendu de monnaie : rendre la monnaie avec le minimum de pièces

Etant donné un système de monnaie à valeurs entières (*) et une somme entière à rendre, on appelle problème du rendu de monnaie le problème qui consiste à **rendre la monnaie avec le moins de pièces possibles**.

(*) *on suppose aussi que le système de monnaie contient la pièce 1 pour être certain de pouvoir rendre la monnaie dans tous les cas*

1.0.5 Synthèse du vocabulaire et des notations utilisées dans la suite du notebook

Dans la suite nous n'utiliserons que les deux systèmes de monnaie et nous appelerons donc systématiquement :

- `systeme_monnaie_euro` le tableau d'**entiers** `[1, 2, 5, 10, 20, 50, 100, 200, 500]`,
- `systeme_monnaie_pluton` le tableau d'**entiers** `[1, 3, 6, 12, 24, 30]`,
- `somme_a_rendre` le montant **entier** de la somme qui doit être rendue (ci-dessus égale à 49),
- `liste_rendu` le tableau des pièces qui vont être rendues (si on s'y prend mal, `liste_rendu` peut utiliser plus de pièces que le minimum possible).

1.0.6 Pour vérifier si on a bien compris

Question :

Pour le systeme `systeme_monnaie_euro` et pour `somme_a_rendre = 37`, trouver :

- un rendu de monnaie `liste_rendu` utilisant selon vous le moins de pièces possibles,
- un rendu de monnaie `liste_rendu` utilisant plus de pièces que le minimum,
- laquelle des deux réponses est appelée "la meilleure solution" ?

Question :

Pour le systeme `systeme_monnaie_pluton` et pour `somme_a_rendre = 37`, trouver :

- un rendu de monnaie `liste_rendu` utilisant selon vous le moins de pièces possibles,
- un rendu de monnaie `liste_rendu` utilisant plus de pièces que le minimum,
- laquelle des deux réponses est appelée "la meilleure solution" ?

2 L'algorithme naturel du rendu de monnaie est un algorithme glouton

Question :

Dans le problème du rendu de monnaie :

- Quelle est la sélection que l'on effectue ?
- Quelle est la contrainte à vérifier par la sélection ?
- Quelle est la maximisation ou minimisation recherchée ?

Pour rendre la monnaie la méthode que tout le monde utilise est la suivante :

```
liste_rendu = liste vide
Tant que somme_a_rendre > 0:
    - choisir la plus grande pièce de systeme_monnaie inférieure à somme_a_rendre
    - mettre cette pièce dans liste_rendu
    - diminuer somme_a_rendre de la valeur de la pièce
```

Question :

Quelle est, à chaque étape, la règle de choix ?

2.0.1 Conclusion

L'algorithme ci-dessus, appelé "algorithme du rendu de monnaie" est bien un algorithme glouton.

3 Programmation version 1

Question :

Programmer une fonction `plus_grande_piece_dans` qui prend en paramètre :

- un tableau d'entiers strictement positifs `systeme_monnaie` qui contient au moins la valeur 1
- un nombre entier `somme` strictement supérieur à 0

et renvoie la plus grande valeur `piece` présente dans `systeme_monnaie` qui est inférieure ou égale à `somme`.

.

Quelques assertions qui doivent être vérifiées par votre fonction sont données ci-dessous.

[ ]:
```
def plus_grande_piece_dans(systeme_monnaie, somme):
    #code à compléter


    return piece
```

[ ]:
```
systeme_monnaie_euro = [1, 2, 5, 10, 20, 50, 100, 200, 500]
systeme_monnaie_pluton = [1, 3, 6, 12, 24, 30]

assert(plus_grande_piece_dans(systeme_monnaie_euro, 23) == 20)
assert(plus_grande_piece_dans(systeme_monnaie_euro, 259) == 200)
assert(plus_grande_piece_dans(systeme_monnaie_euro, 9) == 5)
assert(plus_grande_piece_dans(systeme_monnaie_euro, 1) == 1)

assert(plus_grande_piece_dans(systeme_monnaie_pluton, 23) == 12)
assert(plus_grande_piece_dans(systeme_monnaie_pluton, 259) == 30)
assert(plus_grande_piece_dans(systeme_monnaie_pluton, 9) == 6)
assert(plus_grande_piece_dans(systeme_monnaie_pluton, 1) == 1)
```

Question :

En utilisant la fonction `plus_grande_piece_dans` définis ci-dessus, compléter le code de la fonction `rendre monnaie` qui prend en paramètre :

- un tableau d'entiers strictement positifs `systeme_monnaie` qui contient au moins la valeur 1
- un nombre entier `somme_a_rendre` strictement supérieur à 0

et renvoie la liste `liste_rendu` obtenue par l'algorithme du rendu de monnaie sur `somme_a_rendre`.

On rappelle que pour ajouter un élément `elt` à une liste `L` on utilise l'instruction `L.append(elt)`.

.

Quelques assertions qui doivent être vérifiées par votre fonction sont données ci-dessous.

[ ]:
```
def rendre_monnaie(systeme_monnaie, somme_a_rendre):
    liste_rendu = []
    #code à compléter


    return liste_rendu
```

[ ]:
```
systeme_monnaie_euro = [1, 2, 5, 10, 20, 50, 100, 200, 500]
systeme_monnaie_pluton = [1, 3, 6, 12, 24, 30]

assert(rendre_monnaie(systeme_monnaie_euro, 23) == [20, 2, 1])
assert(rendre_monnaie(systeme_monnaie_euro, 259) == [200, 50, 5, 2, 2])
assert(rendre_monnaie(systeme_monnaie_euro, 9) == [5, 2, 2])
assert(rendre_monnaie(systeme_monnaie_euro, 1) == [1])

assert(rendre_monnaie(systeme_monnaie_pluton, 23) == [12, 6, 3, 1, 1])
assert(rendre_monnaie(systeme_monnaie_pluton, 259) == [30, 30, 30, 30, 30, 30,␣
↪30, 30, 12, 6, 1])
assert(rendre_monnaie(systeme_monnaie_pluton, 9) == [6, 3])
assert(rendre_monnaie(systeme_monnaie_pluton, 1) == [1])
```

Questions :

1) Avec le système `systeme_monnaie_pluton`, donner la `liste_rendu` renvoyée par l'algorithme du rendu de monnaie pour chacune des `somme_a_rendre` ci-dessous :

- 48
- 49
- 50
- 51
- 52
- 53
- 54

2) Montrer que dans les six premiers cas il existe une meilleure solution que celle renvoyée par l'algorithme (qui utilise une pièce de moins).

3) Pour votre culture générale, sachez :

- qu'avec `systeme_monnaie_euro`, la `liste_rendu` renvoyée par l'algorithme du rendu de monnaie est toujours la meilleure solution : on dit que le système de monnaie est *canonique*.
- qu'avec `systeme_monnaie_pluton`, ce n'est pas le cas : on dit que le systeme de monnaie n'est *pas canonique*.

4 Programmation version 2 (facultatif)

Question :

Essayer de comprendre la version 2 ci-dessous de l'algorithme du rendu de monnaie afin d'expliquer pourquoi cette variante est plus efficace que la version 1.

[ ]:
```
def rendre_monnaie_v2(systeme_monnaie, somme_a_rendre):
    liste_rendu = []
    indice_plus_grande_piece = len(systeme_monnaie)-1

    while somme_a_rendre > 0:
        piece = systeme_monnaie[indice_plus_grande_piece]
        if piece <= somme_a_rendre:
            liste_rendu.append(piece)
            somme_a_rendre = somme_a_rendre - piece
        else:
            indice_plus_grande_piece = indice_plus_grande_piece - 1
    return liste_rendu
```

[ ]:
```
systeme_monnaie_euro = [1, 2, 5, 10, 20, 50, 100, 200, 500]
systeme_monnaie_pluton = [1, 3, 6, 12, 24, 30]

assert(rendre_monnaie_v2(systeme_monnaie_euro, 23) == [20, 2, 1])
assert(rendre_monnaie_v2(systeme_monnaie_euro, 259) == [200, 50, 5, 2, 2])
assert(rendre_monnaie_v2(systeme_monnaie_euro, 9) == [5, 2, 2])
assert(rendre_monnaie_v2(systeme_monnaie_euro, 1) == [1])

assert(rendre_monnaie_v2(systeme_monnaie_pluton, 23) == [12, 6, 3, 1, 1])
assert(rendre_monnaie_v2(systeme_monnaie_pluton, 259) == [30, 30, 30, 30, 30,␣
↪30, 30, 30, 12, 6, 1])
assert(rendre_monnaie_v2(systeme_monnaie_pluton, 9) == [6, 3])
assert(rendre_monnaie_v2(systeme_monnaie_pluton, 1) == [1])
```