TP INFORMATIQUE
Algorithmes gloutons

Rendu de monnaie

Le principe

Un achat dit en espèces se traduit par un échange de pièces et de billets. Dans la suite de ce T.P., les pièces désignent indifféremment les véritables pièces que les billets.

Lors d'un rendu de monnaie on cherche souvent à le faire avec un minimum de pièces. C'est le *problème du rendu de monnaie* dont la solution dépend du *système de monnaie* utilisé.

> Par exemple, dans le système monétaire français, les « pièces » prennent les valeurs 1, 2, 5, 10, 20, 50, 100, 200, 500 euros.
> Pour simplifier, nous nous intéressons seulement aux valeurs entières (sinon on transforme en centimes...)
> Rendre de la monnaie avec un minimum de pièces est un *problème d'optimisation*.

En pratique tout individu utilise naturellement un algorithme glouton.

Principe de l'algorithme glouton : chercher la pièce de plus grande valeur possible puis réitérer le procédé.

Modélisation

Tous les nombres manipulés dans la suite sont supposés entiers.

On considère un système de pièces de monnaies de valeurs $v_0 < v_2 < \cdots < v_{n-1}$ placées **par ordre croissant** dans une liste python `v`.

On note `m` le montant à rendre.

On veut récupérer une liste `nb_pieces` de même taille que `v` comportant *le nombre de pièces* de chaque valeur à rendre.

> Par exemple pour rendre 49 euros dans le système français `v = [1,2,5,10,20]` on aura `nb_pieces = [0,2,1,0,2]` (deux fois 2 euros, une fois 5 euros et deux fois 20 euros).

Le travail à effectuer

Exercice 1. Fonction python

> **Conseil.** Pour les questions suivantes réfléchir (sur feuille) aux variables auxiliaires dont vous aurez besoin et aux actions que vos fonctions devront effectuer.
>
> **Remarque.** plusieurs manières de programmer sont possibles...

1. Écrire une fonction Python `indice_piece_max(m,v)` qui retourne l'indice de la plus grande pièce de la liste `v` inférieure au montant `m`
   Par exemple pour `v = [1,2,5,10,20]` *et* $m = 7$*, la fonction retournera l'indice 2.*
   En effet `v[2] = 5` *est la plus grande pièce inférieure à* m*.*
2. Ecrire une fonction Python `rendu_glouton(s, v)` qui retourne la liste `nb_pieces` à partir de la somme `s` et de la liste `v` des valeurs disponibles des pièces.
   On suppose que le nombre de pièces de chaque valeur est illimité.

Exercice 2. Un algorithme pas toujours correct

On peut s'interroger sur l'aspect optimal d'une solution donnée par l'algorithme glouton.

1. Avez-vous bien pensé à tester votre fonction ?! (avec l'exemple donné plus haut ou un autre)
2. Tester maintenant avec le rendu de 49 dans le système $[1, 3, 6, 12, 24, 30]$.
   La solution obtenue est-elle optimale ?

> Lorsque la solution obtenue est toujours optimale, le système monétaire est dit *canonique*.
> Pour savoir si le système est canonique, un algorithme (de Kozen et Zaks) apporte une réponse efficace. Le sujet d'informatique de Centrale 2002 l'abordait. C'était à l'époque avec un autre langage mais on pourrait imaginer une version remise au goût du jour...

Sélection de cours

Le principe

Dans un lycée on veut associer, à une salle donnée, le plus grand nombre de cours possible sur une journée.

Bien sûr dans la salle ne pourra avoir lieu qu'un seul cours la fois... Ceci dit, il n'y a pas de temps de battement entre deux cours ; le cours suivant peut commencer exactement lorsque le précédent termine.

Lorsqu'on vient de choisir un cours, il faut choisir un autre cours parmi ceux non encore sélectionnés et qui ait un horaire compatible.

Pour ce choix on va appliquer le principe suivant.

Principe de l'algorithme glouton : chercher le cours « compatible » terminant le plus tôt puis réitérer le procédé.

Modélisation

Chacun de nos cours sera modélisé par un couple d'entiers (d, f) où d est l'heure de début et f est l'heure de fin. L'ensemble des cours sera modélisée par une liste de tels couples
`liste_cours = [(d0,f0), (d1, f1), ... ]`

Remarque. On supposera que les cours sont déjà triés par horaires de fin croissant.

On veut récupérer la liste `indices_cours` des indices de cours à attribuer à notre salle.

Par exemple, avec cette liste,

indice k	0	1	2	3	4	5	6	7	8	9	10
début d_k	1	3	2	5	3	5	6	8	8	2	12
fin f_k	4	5	6	7	8	9	10	11	12	13	14

,

la fonction retournera `[0, 3, 7, 10]`

Le travail à effectuer

Exercice 3. Fonction python

1. Réfléchissez (sur feuille) aux variables auxiliaires dont vous aurez besoin et aux actions que votre fonction devra effectuer.
   Vous pouvez écrire des fonctions auxiliaires si vous le souhaitez.
   Remarque. plusieurs manières de programmer sont possibles...
2. Écrire une fonction Python `selection_cours` qui retourne la liste `indices_cours` à partir de la liste `liste_cours` des cours *triés par heure de fin croissante.*