TP 22 - Algorithmes gloutons

On étudie une méthode (parmi d'autres) de résolution des problèmes d'optimisation : les **algorithmes gloutons** (*greedy algorithm*). Le principe est de trouver une solution qui à chaque étape intermédiaire fait le choix qui semble le meilleur.

- "Aveuglé par son appétit", le glouton trouve une solution rapidement, mais la solution finale n'est pas forcément optimale.
- Chaque choix est traité par une analyse locale (instantanée) du problème et ramène à un problème plus petit : c'est une progression descendante.
- Il n'y a pas de retour en arrière en vue de tester d'autres possibilités

Problème de rendu de monnaie

Comment rendre une somme donnée avec un minimum de pièces et billets ?

Les hypothèses de simplifications :

- les sommes traitées sont des entiers
- les stocks des pièce et billets sont suffisants
- on désigne par "pièces" à la foi les pièces et les billets

Exemple

Le rendu de monnaie sur 49 euros se décompose, selon une approche gloutonne, par $49 = 20 + 20 + 5 + 2 + 2$.
Ce qui fait 5 pièces.

Exercice 1

1. Donner un algorithme, puis une fonction python itérative, `monnaie(x,P=[500,200,100,50,20,10,5,2,1])` qui réalise le rendu de monnaie de x selon les valeurs de la liste `P` décroissante.
2. Donner une version récursive.

Remarque – On dit qu'un système monétaire est canonique si pour toute somme, l'algorithme glouton conduit à une solution optimale.
Il n'existe pas de caractérisation des systèmes monétaires canoniques, seulement des techniques de vérifications.
L'euro est canonique.

Exercice 2 Montrer que le système impérial britanique (avant 1971) n'est pas canonique :

- 240 pence (livre)
- 60 pence (couronne)
- 30 pence (florin)
- 24 pence (shilling)
- 12 pence (demi-shilling)
- 6 pence
- 3 pence
- 1 penny

Problème de sélection d'activités

Comment organiser des activités afin de pouvoir en faire le maximum ?

Les hypothèses de simplifications :

- les activités ne peuvent se dérouler en parallèle : c'est la même personne qui les fait.
- les activités sont référencées par un triplet : (nom, heure de début, heure de fin)

La stratégie glouton consiste à

(i) classer les activités par heures de fin croissantes
(ii) choisir la première activité de la liste
(iii) choisir parmi celles suivantes, la première compatible (heure de début postérieure à celle de fin de la première)
(iv) recommencer tant que la liste n'a pas été entièrement parcourue

Exercice 3 Proposer une fonction `activites(L)` qui réalise l'algorithme glouton sur la liste d'activités `L`.

On pourra utiliser l'instruction `sorted(T, key=lambda t: t[2])` pour trier la liste `T` par rapport à la 3ième composante des éléments de `t`. Par exemple :

```
sorted([[0,3],[2,2],[1,4],[3,0]],key=lambda t:t[1])
```

qui retourne

```
[[3, 0], [2, 2], [0, 3], [1, 4]]
```

Problème d'allocation de salles pour des cours

Comment allouer des salles pour des cours ?

Les hypothèses de simplifications :

- les cours sont référencées par un triplet : (nom, heure de début, heure de fin)
- le nombre de salles est suffisant, mais l'objectif est d'en utiliser le moins possible

Exercice 4 Proposer une solution et la mettre en oeuvre ?

Problème du sac à dos

Comment optimiser le remplissage d'un sac à dos en fonction de deux critères quantifiés : la valeur (ou l'utilité) et le poids ?

Exercice 5 Proposer plusieurs approches et les tester ?