ISM MPSI/PCSI, Informatique
Année 2025/2026

TP Informatique 13

⚠ On rappelle qu'un script (fichier `*.py`) doit être enregistré et exécuté (touche `F5`) pour que les fonctions saisies dans le script soient utilisables dans la console et que la combinaison de touches `Ctrl+C` permet de casser une boucle infinie.

Exercice 1

On s'intéresse ici au problème de l'allocation de ressources.
Par exemple, dans un lycée, on souhaite allouer différentes salles pour que s'y tiennent des cours.
On peut représenter un cours par l'intervalle $[deb, fin[$, que l'on codera par un triplet de la forme `(deb,fin,'cours')` et dont les extrémités sont les moments de début et de fin. Deux cours dont les intervalles s'intersectent ne peuvent avoir lieu dans la même salle.
Combien de salles sont nécessaires, et quels cours attribuer à chaque salle ?
On va suivre pour cela une approche «gloutonne». Étant donnée une liste d'intervalles représentant les cours, que l'on suppose déjà triée (plusieurs critères sont possibles, on retient ici le tri par heures de fin croissantes) :

- on réserve une salle et on lui affecte le premier cours ;
- pour chaque cours qui doit se tenir :
  - ▷ s'il est possible de le faire se tenir dans une salle déjà utilisée (parce que son moment de début est postérieur à la fin du dernier cours qui se tient dans cette salle), alors on affecte ce cours à cette salle ;
  - ▷ sinon on réserve une nouvelle salle et on lui affecte le cours.

Par exemple, si l'on doit placer les cours (0,4,'maths'), (2,6,'physique'), (6,10,'SI') et (8,11,'lettres') :

— On ouvre une première salle et on lui affecte le cours (0,4,'maths') et on indique que la salle est disponible à partir du moment 4.
— Le second cours débute au moment 2, la première salle n'étant pas libre, on en ouvre une nouvelle pour y affecter le cours et on indique que la salle est disponible à partir du moment 6.
— Le troisième cours débute au moment 6, la première salle est disponible, on affecte donc le cours à la première salle et on met à jour son moment de disponibilité à 10.
— Enfin le dernier cours débute au moment 8, la première salle n'est pas disponible mais la deuxième l'est, donc on lui attribue ce dernier cours et on met à jour son moment de disponibilité.

Si le nombre de cours est grand il devient pertinent de programmer cet algorithme.

On complètera le fichier `TP13.py` disponible sur le site cahier-de-prepa de la classe.

1. Écrire une fonction `est_libre(dispo_salle,deb)` prenant en entrée une liste de nombres `dispo_salle` (représentant les moments à partir desquels les salles déjà réservées sont libres), et un nombre `deb` (le moment où commence le prochain cours à affecter), et qui renvoie l'indice i d'une salle pouvant accueillir le cours s'il en existe au moins une, et -1 sinon.

2. En déduire une fonction `allocation(cours)` prenant en entrée une liste de triplets (deb,fin,'cours') supposés déjà triés dans l'ordre croissant de leur deuxième composante, et renvoyant une liste de listes de triplets, chaque liste représentant une salle contenant les cours l'occupant.
3. Tester le programme avec l'exemple du fichier.

Exercice 2

On s'intéresse dans cet exercice à la décomposition d'un rationnel de $[0, 1[$ en somme de *fractions égyptiennes*.
Une fraction égyptienne est un nombre rationnel r qui peut s'écrire sous la forme $r = \frac{1}{n}$ où $n \in \mathbb{N}^*$.
On admet le théorème suivant : un rationnel positif x (non nul) peut toujours s'écrire comme somme de fractions égyptiennes deux à deux distinctes.
Par exemple : $\frac{2}{3} = \frac{1}{2} + \frac{1}{6}$ mais aussi $\frac{2}{3} = \frac{1}{3} + \frac{1}{6} + \frac{1}{9} + \frac{1}{18}$. Autrement dit la décomposition d'un rationnel en somme de fractions égyptiennes n'est pas unique.

Si l'on se restreint à des rationnels x de $[0, 1[$, on dispose d'une stratégie gloutonne (déjà connue de Fibonacci au XIIIè siècle) pour déterminer une décomposition de x en somme de fractions égyptiennes : on initialise une variable r à la valeur de $x \in [0, 1[$ et un dénominateur k à 1, puis tant que $r \neq 0$, on incrémente k jusqu'à ce que $\frac{1}{k} \leqslant r$ et on remplace alors r par $r - \frac{1}{k}$.
On peut démontrer à l'aide d'une récurrence forte sur le numérateur de x que cet algorithme fournit bien une décomposition en somme de fractions égyptiennes de x.

1. Écrire une fonction `fraction(a,b)` prenant en argument deux entiers naturels a, b tels que $\frac{a}{b} \in [0, 1[$ et renvoyant une décomposition en somme de fractions égyptiennes de $\frac{a}{b}$ (sous la forme d'une chaîne de caractères).
   Tester la fonction avec $\frac{1}{3}$, avec $\frac{5}{16}$, avec $\frac{2}{5}$.
2. A priori la fonction précédente débouche sur une boucle infinie avec la dernière valeur de la question précédente. En expliquer la cause, et proposer une amélioration de la fonction.