ISM MPSI/PCSI, Informatique
Année 2024/2025

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 $[deb, fin[$ dont les extrémités sont les moments (heures) 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 au minimum, et quels cours attribuer à chaque salle ?
On va suivre pour cela l'algorithme glouton suivant :

```
Entrée : Un ensemble I d'intervalles triés par date de fin croissante (les cours)
Sortie : Un ensemble minimal de salles pour accueillir les cours
S <- [] ; # liste vide de salles
Pour tout intervalle J=[deb; fin[ de I faire :
    S'il existe une salle de S inoccupée au moment deb alors :
        L'ajouter à cette salle
    Sinon :
        Créer une autre salle contenant uniquement J
Renvoyer S
```

L'algorithme affecte les cours dans des salles de sorte que deux cours n'ont jamais lieu en même temps dans la même salle.

1. Écrire une fonction `est_libre(T,deb)` prenant en entrée une liste de nombres `T` (représentant les moments où les salles déjà créées sont libres), et un nombre `deb` (l'heure 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(I)` prenant en entrée une liste de couples (les intervalles, supposés déjà triés dans l'ordre croissant de leur deuxième composante) et renvoyant une liste de listes de couples, chaque liste représentant une salle contenant les cours l'occupant.
3. Tester le programme avec la liste des cours

$$\mathrm{L} = [(0,3),(1,5),(5,8),(4,9),(4,10),(11,19),(8,20),(12,21),(10,23),(22,27),(25,30)]$$

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.