TP noté

PTSI Lycée Eiffel

17 décembre 2015

Quelques remarques avant de commencer :

- **Les notes de cours sont autorisées (mais PAS le recours à des programmes tout faits trouvés sur Internet).**
- **Les seules fonctions Python sur les listes que vous avez le droit d'utiliser sont *len* et la méthode *append*. En particulier, PAS de *sort* ou de *sum*.**
- **Les étudiants rendront à la fin du TP une copie papier (certaines questions nécessitent d'être traitées sur papier) ainsi qu'un fichier Python (extension .py) contenant leurs programmes que l'enseignant récuperera directement sur une clé USB. Il est bien entendu autorisé et même conseillé de commenter les programmes écrits sur ce fichier (les commentaires en Python s'introduisent avec un #).**

I. Quelques algorithmes élémentaires sur les listes.

Cette première partie est relativement indépendante des autres, même s'il est tout à fait autorisé de réutiliser certaines des fonctions programmées ici dans la suite de l'épreuve. Il s'agit d'écrire en Python quelques algorithmes sur les listes, histoire de s'échauffer un peu.

Question 1 : Écrire une fonction Python **egalite(l,m)** déterminant si deux listes l et m contiennent les mêmes éléments (dans le même ordre). On supposera que les listes sont de même longueur (mais vous pouvez mettre un test à l'intérieur de la fonction pour vérifier que c'est bien le cas).

Exemple : egalite([1,2],[1,3]) doit retourner la valeur False (ou un message indiquant que les listes sont différentes) ; **egalite([1,2],[1,2])** doit retourner True.

Question 2 : Écrire une fonction Python **supprimenegatifs(l)** qui retourne une liste où tous les éléments négatifs de l ont été remplacés par des zéros.

Exemple : supprimenegatifs([2,-3,5,1,-2]) doit retourner la liste [2,0,5,1,0].

Question 3 : Écrire une fonction Python **positifs(l)** qui retourne une liste ne contenant que les éléments positifs (ou nuls) de la liste l (et supprimant les négatifs).

Exemple : positifs([2,-3,5,1,-2]) doit retourner la liste [2,5,1].

Question 4 : Écrire une fonction Python **indices(l,a)** déterminant les indices de la liste où apparait la valeur a. La fonction renverra une liste, vide dans le cas où a n'apparait jamais dans la liste.

Exemple : indices([2,7,1,5,7,7],7) doit retourner la liste [1,4,5].

II. Quelques fonctions sur les couples d'entiers.

Dans toute la suite de cet énoncé, un couple d'entiers positifs (p, q) sera représenté en Python par la liste de deux éléments [p,q]. Ces couples seront eux-mêmes utilisés pour représenter des nombres rationnels, par l'intermédiaire de la fonction *evalue* suivante : pour tout couples d'entiers non nuls (p, q), on pose simplement $evalue(p, q) = \frac{p}{q}$. Si $q = 0$ et $p \neq 0$, on impose $evalue(p, q) = +\infty$ (ce qu'on désignera pas la chaine de caractères « infini » en Python). Enfin, si $p = q = 0$, on convient que $evalue(p, q) = 1$.

Question 5 : Écrire une fonction Python **evalue(l)** qui retourne la valeur donnée par la fonction evalue définie ci-dessus. On supposera que l est une liste de deux nombres entiers, et on gèrera les cas particuliers pour $q = 0$.

Exemple : evalue([1,2]) doit retourner le nombre 0.5.

Si (p, q) et (p', q') sont deux couples d'entiers, le couple médian de ces deux couples est le couple $(p + p', q + q')$.

Question 6 : Écrire une fonction Python **median(l,m)** qui retourne la liste de deux entiers correspondant à la définition ci-dessus.

Exemple : median([1,2],[3,5]) doit retourner la liste [4,7].

Question 7 : Écrire une fonction Python **egalitex(x,l)** qui s'applique à un nombre flottant x et à une liste de deux entiers [p,q], et qui renvoie True si $evalue(p, q) = x$ (et False sinon, y compris dans les cas où evalue donne la valeur infini).

Exemple : egalite(2,[4,2]) doit retourner la valeur True.

Question 8 : Écrire une fonction Python **compare(x,l)** sur le même modèle que la précédente et qui renvoie True si $x \leqslant evalue(l)$, en considérant que tout flottant est plus petit que infini.

Exemple : compare(2,[2,4]) doit retourner False.

III. Suites de Farey.

Les suites de Farey (F_n) sont des listes de couples d'entiers définies par récurrence de la façon suivante :

- F_0 est la liste $[[0, 1], [1, 0]]$.
- Si $F_n = [c_1, c_2, \ldots, c_p]$, alors $F_{n+1} = [c_1, m_1, c_2, m_2, \ldots, c_{p-1}, m_{p-1}, c_p]$, où m_i désigne le couple médian (au sens de la question 6) des couples c_i et c_{i+1}.

Question 9 : Écrire une fonction Python **inseremedians(l)** qui prend comme argument une liste l de couples d'entiers (soit $l = [c_1, \ldots, c_p]$), et renvoit la liste de couples $[c_1, m_1, \ldots, m_{p-1}, c_p]$, où, comme décrit ci-dessus, m_i est le médian de c_i et c_{i+1}. En déduire une fonction **Farey(n)** qui prend comme argument un entier n et renvoit la suite de Farey F_n.

Exemple : Farey(2) doit retourner la liste [[0,1],[1,2],[1,1],[2,1],[1,0]].

On fait la conjecture suivante : dans la suite de Farey (F_n), les valeurs prises par la fonction evalue sur chacun des couples de la suite sont strictement croissantes.

Question 10 : Écrire une fonction Python **evalfarey(n)** qui retourne la liste des valeurs $evalue(c_i)$ pour tous les couples c_i constituant la suite de Farey F_n. En déduire une fonction **conjecture(n)** qui renvoie la valeur True si notre conjecture est vérifiée sur la suite de Farey F_n.

On effectue la deuxième conjecture suivante : en notant $F_n = [c_1, \ldots, c_p]$, et $c_i = [x_i, y_i]$ pour chacun des couples constituant la suite, on a toujours $x_i y_{i+1} - x_{i+1} y_i = -1$ (pour tout entier i compris entre 1 et $p-1$).

Question 11 : Sur le modèle de la question précédente, écrire un programme permettant de tester cette conjecture sur la suite de Farey F_n.

IV. Rercherche dichotomique.

Dans cette dernière partie, x est un nombre flottant positif quelconque.

Question 12 : Écrire une fonction Python **appartientfarey(x,n)** qui détermine si le nombre flottant x correspond à la valeur $evalue(c_i)$ pour un des couples d'entiers c_i de la suite de Farey F_n. La fonction renverra bien sûr True si c'est le cas, False sinon.

On va maintenant présenter un algorithme de recherche de x dans les suites de Farey nettement plus efficace, basé sur une approche dichotomique. On pose initialement $g = [0, 1]$ et $d = [1, 0]$ (autrement dit les deux listes constituant la suite de Farey F_0), puis, à chaque étape de l'algorithme, on calcule le médian m des couples g et d. Si $evalue(m) < x$, on remplace g par m, sinon on remplace d par m. On admet qu'en procédant de cette façon, on aura toujours $evalue(g) \leqslant x \leqslant evalue(d)$, et qu'après n étapes de l'algorithme, g et d seront deux couples consécutifs de la suite de Farey F_n, et que les valeurs de la fonction evalue prises sur ces couples forment le meilleur encadrement possible de x par deux éléments de cette suite.

Question 13 : Écrire une fonction Python **dichofarey(x,n)** qui effectue le calcul de g et de d après n étapes de cet algorithme. En déduire une fonction **appartientfareybis(x,n)** qui affiche tous les médians calculés lors des étapes de cet algorithme, et qui renvoie à la fin True seulement si $evalue(g) = x$ ou $evalue(d) = x$ (ce qui revient à dire que x appartient à F_n).

Question 14 : Écrire une fonction Python **approxfarey(x,e)** qui retourne le plus petit entier n pour lequel la distance entre $evalue(g)$ et $evalue(d)$ est inférieure à e (précision de l'approximation souhaitée), ainsi que les deux couples g et d correspondants.

Exemple : On donnera ainsi les couples permettant d'approcher $\sqrt{2}$ à 10^{-5} près par deux termes consécutifs d'une suite de Farey.