Contrôle d'informatique

Durée : 1 heure

Ce contrôle est constitué de deux exercices indépendants.

Rappel sur les listes

Une suite finie comme par exemple $2, 3, 5, 7, 11, 13$ est le plus souvent représentée en Python par une liste `lst = [2, 3, 5, 7, 11, 13]`. La suite vide sera naturellement représentée par la liste vide `[]`.

– Chaque élément d'une liste est accessible (et modifiable) par le biais de son *index*, numéroté à partir de 0. Par exemple `lst[0]` renvoie la valeur 2, `lst[1]` renvoie la valeur 3, `lst[2]` renvoie la valeur 5, etc.

– La fonction `len` renvoie la longueur de la liste ; par exemple, `len(lst)` renvoie la valeur 6. Les index d'une liste sont donc compris (au sens large) entre 0 et `len(lst)-1`.

– La méthode `append` permet d'ajouter un élément en fin de liste. Par exemple, après l'instruction `lst.append(17)`, la liste `lst` est désormais de longueur 7 et vaut `[2, 3, 5, 7, 11, 13, 17]`.

Exercice 1 Fractions égyptiennes

On appelle *fraction égyptienne* une somme de fractions unitaires, c'est-à-dire qui ont des numérateurs égaux à 1 et des dénominateurs entiers supérieurs ou égaux à 2, ces dénominateurs étant deux-à-deux distincts.
Par exemple, $\frac{2}{5}$ est une fraction égyptienne puisqu'on peut écrire : $\frac{2}{5} = \frac{1}{5} + \frac{1}{6} + \frac{1}{30}$, mais aussi $\frac{2}{5} = \frac{1}{3} + \frac{1}{15}$.
Il est possible de démontrer que tout rationnel de l'intervalle $]0, 1[$ peut être représenté sous la forme d'une fraction égyptienne, et qu'il peut être décomposé d'une infinité de façons différentes.

Dans cet exercice, on choisit de représenter un nombre rationnel p/q par un couple `(p, q)`, sans se préoccuper de caractère irréductible ou non de la représentation. Par exemple, le rationnel 2/5 pourra tout aussi bien être représenté par le couple `(2, 5)` que par le couple `(6, 15)`.
La fraction égyptienne $\frac{1}{a_1} + \frac{1}{a_2} + \cdots + \frac{1}{a_n}$ sera représentée par la liste `[a1, a2, ... , an]`, *en imposant en outre la condition* $2 \leqslant a_1 < a_2 < \ldots < a_n$, condition qui assure que les dénominateurs sont deux-à-deux distincts.
Là encore, il n'y a pas unicité de la représentation, puisque le rationnel 2/5 peut aussi bien être représenté sous forme de fraction égyptienne par la liste `[5, 6, 30]` que par la liste `[3, 15]`.

a) Rédiger une fonction `estEgyptienne(lst)` qui prend en argument une liste d'entiers naturels strictement positifs `lst` et qui renvoie le booléen `True` lorsque `lst` représente une fraction égyptienne, et `False` sinon.

Par exemple, `estEgyptienne([2, 3, 3, 5])` et `estEgyptienne([1, 3, 7, 8])` renverrons dans les deux cas le booléen `False` puisque la première liste n'est pas strictement croissante et la seconde débute par 1.

b) Rédiger une fonction `rationnel(lst)` qui prend en argument une liste `lst` *supposée représenter une fraction égyptienne* et qui renvoie un couple (p, q) représentant le rationnel décrit par cette fraction égyptienne.
Par exemple, `rationnel([5, 6, 30])` pourra tout aussi bien renvoyer le couple $(2, 5)$ que le couple $(6, 15)$.

On considère l'algorithme suivant, rédigé en Python :

```
def fractionEgyptienne(p, q):
    a = p
    b = q
    lst = []
    while b % a != 0:
        m = b // a + 1
        lst.append(m)
        a = a * m - b
        b = b * m
    lst.append(b // a)
    return lst
```

c) Détailler l'exécution de cette fonction pour le couple $(p, q) = (19, 20)$.

On note a_k et b_k les valeurs prises par `a` et `b` à l'entrée de la boucle de rang k, et $L_k = [m_0, \ldots, m_{k-1}]$ l'état de la liste `lst` à l'entrée de cette même boucle. Pour $k = 1$ on a donc $a_1 = p$, $b_1 = q$ et $L_1 = [\,]$.

d) À quelle condition, portant sur a_k et b_k, les valeurs m_k, a_{k+1} et b_{k+1} sont-elles définies ? Lorsque c'est le cas, exprimer les valeurs m_k, a_{k+1} et b_{k+1} en fonction de a_k et b_k.

e) Justifier que $1 \leqslant a_{k+1} < a_k$, et en déduire la terminaison de l'algorithme.

f) Calculer $\frac{a_k}{b_k} - \frac{a_{k+1}}{b_{k+1}}$, puis montrer que le résultat renvoyé lorsque $p < q$ est une représentation du rationnel $\frac{p}{q}$ sous la forme d'une fraction égyptienne.

g) Majorer en fonction de p et q la longueur de la représentation sous forme de fraction égyptienne que renvoie cette fonction. Justifier que ce majorant est atteint lorsque $p = n$ et $q = n! + 1$ (**difficile**).

Rappel sur les chaînes de caractères

Un mot vu comme une suite finie de lettre est représenté en Python par une chaîne de caractères, comme par exemple `chn = 'azertyuiop'`. Le mot vide est naturellement représenté par la chaîne vide `''`.

– Chaque caractère individuel est accessible par le biais de son *index*, numéroté à partir de 0. Par exemple `chn[0]` renvoie le caractère `'a'`, `chn[1]` renvoie le caractère `'z'`, `chn[2]` renvoie le caractère `'e'`, etc.

– La fonction `len` renvoie la longueur d'une chaîne ; par exemple, `len(chn)` renvoie la valeur 10. Les index d'une chaîne sont donc compris (au sens large) entre 0 et `len(chn)-1`.

– Si `s` représente la chaîne $s_0 s_1 \cdots s_{n-1}$ et $0 \leqslant i \leqslant j \leqslant n$, `s[i:j]` désigne la chaîne $s_i s_{i+1} \cdots s_{j-1}$; `s[i:]` est équivalent à `s[i:len(s)]` et `s[:j]` est équivalent à `s[0:j]`.

Exercice 2 Recherche d'un motif dans un mot

Dans cet exercice, on considère des mots u formés à partir d'un ensemble de lettres constituant un alphabet A : $u = u_0 u_1 \cdots u_{p-1}$ avec $u_i \in$ A, $0 \leqslant i \leqslant p-1$. On dit que p est la *longueur* du mot u, et on convient de noter ε le mot vide, c'est-à-dire l'unique mot de longueur nulle.
Si $u = u_0 u_1 \cdots u_{p-1}$ et $v = v_0 v_1 \cdots v_{q-1}$ on convient de noter uv le résultat de la concaténation de ces deux mots, autrement dit le mot $u_0 u_1 \cdots u_{p-1} v_0 v_1 \cdots v_{q-1}$, de longueur $p+q$.

- Un mot v est un *préfixe* de u lorsqu'il existe un mot w tel que $u = vw$;
- un mot v est un *suffixe* de u lorsqu'il existe un mot w tel que $u = wv$;
- un mot v est un *facteur* de u lorsqu'il existe deux mots w_1 et w_2 tels que $u = w_1 v w_2$.

Un préfixe (respectivement suffixe, facteur) de u est dit *propre* lorsque sa longueur est strictement inférieure à celle de u.

Les mots seront représentés en Python par le type `str`.

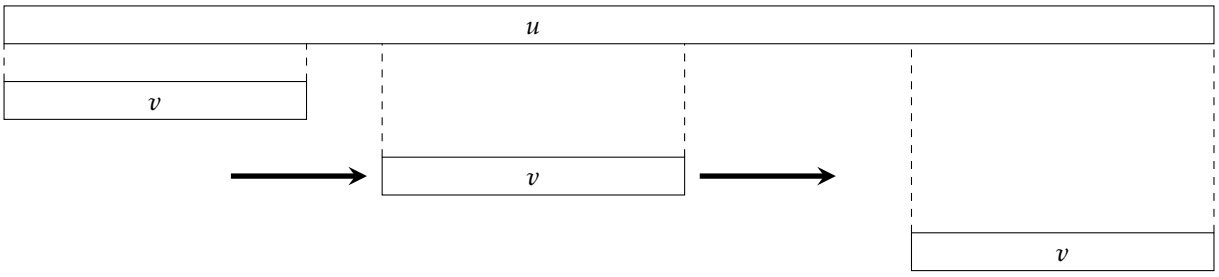
Dans cet exercice, on s'autorise uniquement à comparer deux caractères individuels. Interdiction est donc faite de comparer entre elles deux chaînes de caractères de longueurs supérieures ou égales à 2.

a) Rédiger en Python une fonction `estPrefixe(u, v)` qui prend en arguments deux mots u et v et renvoie `True` lorsque v est préfixe de u et `False` sinon.
Si p est la longueur du mot u et q celle du mot v, exprimer en fonction de p et q le nombre maximal de comparaisons entre caractères effectuées par cet algorithme.

b) On appelle *bord* d'un mot u le plus grand suffixe propre de u qui soit aussi préfixe de u. Ce mot est noté bord(u).
Quel sont les bords des mots *abaababa* et *abababa* ?

Rédiger une fonction `bord(u)` qui prend en argument un mot u et renvoie son bord (utiliser la fonction `estPrefixe`).

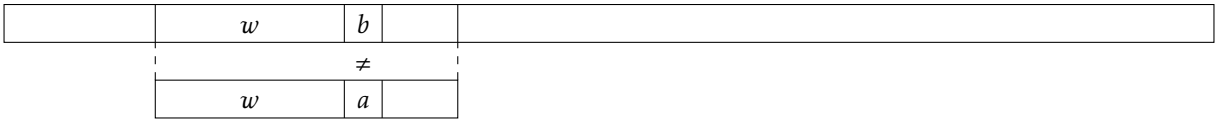
Dans la suite de cet exercice, on souhaite écrire une fonction `estFacteur(u, v)` qui renvoie le booléen `True` lorsque v est facteur de u et `False` sinon. Il existe de nombreux algorithmes résolvant ce problème. La plupart utilisent le concept de *fenêtre glissante* : le mot v est comparé au facteur de u placé en regard, puis, en cas d'échec de la comparaison, la fenêtre contenant le mot v est décalée vers la droite.



c) L'algorithme le plus simple consiste à décaler la fenêtre qu'un cran vers la droite après chaque échec. Rédiger la fonction correspondante `estFacteur(u, v)` qui prend en arguments deux mots u et v et renvoie `True` lorsque v est facteur de u et `False` sinon (là encore, on peut utiliser la fonction `estPrefixe`).

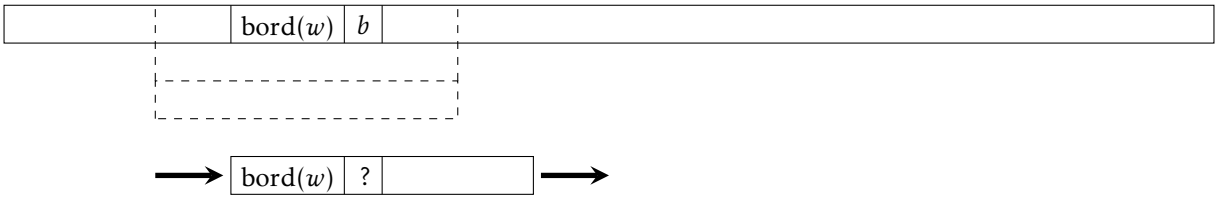
Exprimer en fonction des longueurs p et q des mots u et v le nombre maximal de comparaisons entre caractères effectués par cet algorithme.

d) **(difficile)**. L'algorithme de Morris et Pratt repose sur la constatation suivante. Considérons une situation d'échec lors de la comparaison entre v et un facteur de u :



Le mot wa est un préfixe de v et a est la première lettre qui ait provoqué un échec dans la comparaison entre v et un facteur de u.

Si on connait la longueur du bord de w, on peut décaler la fenêtre de cette longueur et poursuivre la comparaison au delà du bord de w :



On suppose calculées les longueurs des bords de chacun des préfixes de v, ces valeurs étant stockées dans un tableau `bord` : si $k \geqslant 1$, `bord[k]` est la longueur du bord du mot $v_0 v_1 \dots v_{k-1}$ (la case de rang 0 n'est pas utilisée).

Rédiger une fonction `estFacteurMP(u, v, bord)` qui prend en arguments les mots u et v et la tableau `bord` et qui renvoie le booléen `True` lorsque v est facteur de u, et `False` sinon, en suivant l'algorithme de Morris et Pratt.

♦ ♦
♦