Banque « Agro- Veto »
AT - 0519

Algorithmique et Informatique
Durée : 45 mn

Si au cours de l'épreuve, un candidat repère ce qui lui semble être une erreur d'énoncé, il le signale sur sa copie et poursuit sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.
L'usage d'une calculatrice est interdit pour cette épreuve.

Le sujet s'intéresse au problème de détermination de la longueur de la plus longue sous-séquence commune à deux séquences ADN. Cette longueur est un indicateur de proximité d'espèces permettant de les comparer lors d'une étude phylogénétique.
Les parties 1, 2 et 3 sont indépendantes. On pourra utiliser les fonctions de la partie 1 dans la partie 3.

1 Questions préliminaires

a. Écrire une fonction `maximum` qui renvoie le plus grand des deux nombres passés en argument.

Par exemple, `maximum(2, 4)` *devra renvoyer* `4`.

b. Écrire une fonction `zeros` qui prend en argument deux entiers naturels n et p (supposés non nuls) et qui renvoie une liste de n listes contenant chacune p zéros, représentant ainsi la matrice nulle à n lignes et p colonnes.

Par exemple, `zeros(2,3)` *devra renvoyer* `[[0,0,0], [0,0,0]]`.

2 Plus longue sous-chaine commune

On considère $A = a_1...a_n$ et $B = b_1...b_p$ deux chaines de caractères non vides.
On appelle **sous-chaine de** A toute chaine de caractères $a_{i_1}...a_{i_k}$ où $1 \leqslant i_1 < \cdots < i_k \leqslant n$ (**ces caractères ne sont pas nécessairement consécutifs** dans A).
On appelle **plus longue sous-chaine commune** à A et B toute sous-chaine commune à A et B de longueur maximale. Si l'une des chaines A ou B est vide, ou si A et B n'ont aucune sous-chaine commune, on convient que la chaine vide est l'unique plus longue sous-chaine commune à A et B.
On s'intéresse alors au problème ci-dessous.

> *Étant données deux chaines de caractères A et B de longueurs respectives n et p, quelle est la longueur d'une plus longue sous-chaine commune à A et B ?*

Par exemple, les chaines de caractères `"AAA"` *et* `"TAA"` *sont des plus longues sous-chaines communes aux chaines de caractères* `chaine1 = "ATAGA"` *et* `chaine2 ="TAACA"`.
La chaine de caractères `sschaine = "ATGC"` *est une plus longue sous-chaine commune aux chaines de caractères* `chaine1 = "AATGCG"` *et* `chaine2 ="TATTAGC"`.

a. Quelle est la longueur d'une plus longue sous-chaine commune aux chaines `"AATGCG"` et `"TATTAGC"` ? *Justifier.*

b. Déterminer une plus longue sous-chaine commune aux chaines `"AATGCG"` et `"TATTAGC"` autre que `"ATGC"`.

c. Parmi les chaines de caractères ci-dessous, indiquez sur votre copie quelle est la seule plus longue sous-chaine commune aux chaines `"TCGTA"` et `"CTG"` ? *On ne demande pas de justifier la réponse.*

☐`"CTG"` ☐`"TCGT"`
☐`"CGT"` ☐`"CG"`

d. Déterminer toutes plus longues sous-chaines communes aux chaines `"TCGTA"` et `"CTG"`.

On ne demande pas de justifier la réponse.

e. Parmi les fonctions ci-dessous, déterminer les deux seules permettant de déterminer si une chaine de caractères `ssch` est une sous-chaine d'une chaine de caractères `ch`. *On ne demande pas de justifier la réponse.*

```
def estSousChaine1(ch,ssch):
    n = len(ch)
    p = len(ssch)
    i, j = 0, 0
    while i < n and j < p:
        if ch[i] == ssch[j]:
            j += 1
        i += 1
    return j == p
```

```
def estSousChaine2(ch,ssch):
    n = len(ch)
    p = len(ssch)
    i, j = 0, 0
    while i < n and j < p:
        if ch[j] == ssch[i]:
            j += 1
        i += 1
    return i == j
```

```
def estSousChaine3(ch,ssch):
    n, p = len(ch), len(ssch)
    for i in range(n):
        j = 0
        while j < p and ch[i+j]==ssch[j]:
            j += 1
        if j == p:
            return True
    return False
```

```
def estSousChaine4(ch,ssch):
    j = 0
    for i in range(len(ch)):
        if ch[i] == ssch[j]:
            j += 1
        if j == len(ssch):
            return True
    return False
```

f. Écrire alors une fonction booléenne `sousChaineCommune` qui prend en argument trois chaines de caractères `chaine1`, `chaine2` et `sschaine` qui renvoie `True` si `sschaine` est une sous-chaine commune à `chaine1` et `chaine2`, et `False` dans le cas contraire.

3 Recherche d'une solution par programmation dynamique

Si $A = a_1...a_n$ et $B = b_1...b_p$ sont deux chaines de caractères non vides, on note $\ell_{i,j}$ la longueur d'une plus longue sous-chaine commune aux chaines $a_1a_2...a_i$ et $b_1b_2...b_j$ (respectivement composées des i premiers et j premiers caractères de A et B).
On peut montrer que la suite double $(\ell_{i,j})_{(i,j)\in[\![0,n]\!]\times[\![0,p]\!]}$ vérifie l'initialisation et la relation de récurrence :

$$\forall (i,j) \in [\![0,n]\!] \times [\![0,p]\!],\ \ell_{i,j} = \begin{cases} 0 & \text{si } i = 0 \text{ ou } j = 0 \\ 1 + \ell_{i-1,j-1} & \text{sinon si } a_i = b_j \\ \max(\ell_{i-1,j}, \ell_{i,j-1}) & \text{sinon si } a_i \neq b_j. \end{cases}$$

On cherche donc à calculer la longueur d'une plus longue sous-chaine commune à $A = a_1...a_n$ et $B = b_1...b_p$, c'est-à-dire le coefficient $\ell_{n,p}$.
Le principe est de calculer de proche en proche chaque coefficient $\ell_{i,j}$ (en respectant l'ordre défini par la relation de récurrence) en les mémorisant dans une matrice.

a. Quel est la taille (nombres de lignes et de colonnes) de la matrice où on mémorisera les coefficients $\ell_{i,j}$?

b. Compléter en annexe le code de la fonction `lplscc` calculant la longueur d'une plus longue sous-chaine commune à deux chaines de caractères `chaine1` et `chaine2` passées en argument.

c. Donner la matrice des coefficients $(\ell_{i,j})$ dans le cas où `A="CTG"` et `B="TCGT"`.

d. Expliquer (sans l'implémenter) comment adapter l'algorithme donné par la fonction `lplscc` pour construire une plus longue sous-chaine commune.

e. On sait qu'une chaine de n caractères admet au plus 2^n sous-chaines distinctes (en comptant la chaine vide).

Expliquez pourquoi la méthode programmée à la question précédente est plus efficace qu'en comparant chaque sous-chaine de A avec chaque sous-chaine de B.

FIN DU SUJET

Annexe : programme principal

```
def lplsch(chaine1, chaine2):
    n, p = len(chaine1), len(chaine2)
    l = zeros(..... , .....)
    for i in range(..... , n+1):
        for j in range(1, .....):
            if chaine1[i-1] == chaine2[j-1]:
                l[i][j] = .....
            else:
                l[i][j] = maximum(..... , .....)
    return l[.....][.....]
```