Devoir Bilan d'Informatique

PTSI Lycée Eiffel

4 juin 2021

Autour du séquençage du génome.

On s'intéresse dans ce sujet à la recherche d'un motif dans une molécule d'ADN. Une molécule d'ADN est constituée de deux brins complémentaires, qui sont un long enchaînement de nucléotides de quatre types différents désignés par les lettres A, T, C et G. Les deux brins sont complémentaires : « en face » d'un A, il y a toujours un T et « en face » d'un C, il y a toujours un G. Pour simplifier le sujet, on va considérer qu'une molécule d'ADN est représentée par une seule chaîne de caractères ne faisant intervenir que les caractères A, C, G et T (on s'intéresse donc seulement à un des deux brins). On parlera de séquence d'ADN pour désigner une telle chaîne.

Les différentes parties du devoir sont très largement indépendantes (seule la lecture de l'introduction de la partie II est nécessaire pour comprendre l'enjeu des parties III et IV) et peuvent être traitées dans l'ordre de votre choix.

I. Génération d'une séquence aléatoire d'ADN.

On rappelle que le module **random** de Python donne (entre autres) accès aux deux fonctions suivantes :

- la fonction **random()** renvoie un nombre flottant aléatoire compris entre 0 et 1
- la fonction **randint(a,b)** renvoie un nombre entier aléatoire compris entre a et b, tous les deux inclus.

On pourra utiliser librement ces fonctions dans les programmes écrits pour cette première partie, à condition d'écrire explicitement en début de programme la commande permettant l'importation de ces fonctions (ou du module random dans son intégralité).

1. On définit une chaîne de caractère par la commande seq='ACTGTAGCTAG'.
   (a) Que renvoie la commande seq[4] ?
   (b) Que renvoie la commande seq[2 :6] ?
   (c) Que renvoie la commande len(seq) ?

(d) On effectue successivement les opérations L=list(seq), puis L.sort(), L.reverse() et enfin print(str(L)). Quelle chaîne de caractères va s'afficher ?

2. Écrire une fonction Python **seqaleatoire(n)** prenant comme argument un entier n et renvoyant une séquence d'ADN aléatoire de longueur n.
3. Écrire une fonction Python **compteA(s)** prenant comme argument une séquence d'ADN s et renvoyant le nombre de nucléotides de type 'A' de la séquence (il n'est pas demandé de vérifier que la séquence est correcte, c'est-à-dire qu'elle ne contient que les caractères A, C, G et T).
4. Écrire une fonction Python **comptenucleo(s)** prenant comme argument une séquence d'ADN s et renvoyant le nombre de nucléotides de chaque type de la séquence, sous forme d'une liste de quatre entiers.
5. Le génome humain est constitué d'environ 30 000 gènes, chacun d'entre eux correspondant à une séquence d'ADN dont la longueur moyenne est d'envison 100 000 nucléotides. On souhaite stocker l'intégralité de ce génome sur une clé USB d'une capacité de 1Go. En supposant le stockage complètement optimisé (on ne peut rien stocker d'autre que des A, C, G et T), aura-t-on assez de place ?

II. Recherche d'un motif par algorithme naïf.

Un motif d'une chaîne de caractères est simplement une sous-chaîne de la chaîne complète. Ainsi, en prenant l'exemple de la chaîne seq='ACTGTAGCTAG', on peut dire que 'TAG' est un motif de seq (ce motif apparait même deux fois dans la chaîne), mais 'TAGG' n'est pas un motif de seq. L'objectif des prochaines parties du sujet est d'étudier des algorithmes permettant de déterminer si une chaîne de caractères m donnée est un motif au sein d'une chaîne plus longue s.

Le premier algorithme proposé est l'algorithme naïf suivant : on parcourt la chaîne s, et pour chaque entier $i < len(s) - len(m)$, on vérifie si la sous-chaîne de la chaîne s débutant à $s[i]$ et de longueur égale à $len(m)$ est égale au motif recherché m. Si c'est le cas, on arrête immédiatement les vérifications et on renvoie l'indice i correspondant.

1. Pourquoi a-t-on imposé la condition $i < len(s) - len(m)$ dans la description précédente (au lieu de simplement prendre $i < len(s)$) ?
2. Écrire une fonction Python **recherchemotif(s,m)** qui prend comme arguments deux chaînes de caractères s et m et renvoie -1 si m n'est **pas** un motif de s, et l'indice i correspondant à la description précédente si m est un motif de s. Ainsi, recherchemotif('ACTGTAGCTAG','TAG') doit renvoyer la valeur 4, et recherchemotif('ACTGTAGCTAG','TAGG') la valeur -1.
3. Au maximum, combien d'opérations élémentaires l'algorithme précédent devra-t-il effectuer (on considère que les seules opérations élémentaires pertinentes ici sont les comparaisons de caractères) ?
4. Si on veut chercher un motif de 50 caractères dans un génome comprenant 3×10^9 nucléotides, combien de temps faudra-t-il avec un ordinateur capable d'effectuer 10^{12} opérations élémentaires par seconde (une réponse appoximative est suffisante) ?
5. Pour comparer deux génomes contenant environ un milliard de nucléotides chacun, on procède de la façon suivante : on découpe le premier génome en morceaux de 50 nucléotides chacun, puis on cherche chacun de ces morceaux dans le second génome. Estimer grossièrement le temps nécessaire pour effectuer cette opération avec le même ordinateur que pour la question précédente. L'algorithme naïf est-il utilisable en pratique pour ce genre de tâche ?

III. Recherche d'un motif par l'algorithme de Knuth-Morris-Pratt.

L'algorithme naïf présente l'inconvénient de ne pas optimiser les comparaisons de caractères : si par exemple le test effectué pour $i = 0$ (début de la chaîne s) échoue parce que le cinquième caractère de s n'est pas le même que le cinquième caractère du motif m, on dispose déjà d'informations sur les caractères numéro 1, 2 et 3 de la chaîne qui pourraient être exploitées pour tester plus rapidement les valeurs suivantes de i. Pour éviter cela, l'algorithme KMP (Knuth-Morris-Pratt) utilise une fonction auxiliaire qui cherche le plus long **préfixe** d'un motif donné qui en soit aussi un **suffixe**. Un préfixe d'une chaîne de caractères est simplement une sous-chaîne de caractères située au début de la chaîne, et différente de la chaîne tout entière. Ainsi, pour la chaîne 'miaou', on aura pour préfixes 'mi' ou 'miao' mais pas 'iao' (sous-chaîne qui n'est pas située au début de la chaîne), ni 'miaou' (intégralité de la chaîne initiale). De même, un suffixe est une sous-chaîne située à la fin de la chaîne, et distincte de la chaîne tout entière.

1. Donner la liste de tous les préfixes et suffixes de la chaîne 'ATGCGTA'.
2. Quel est le plus long préfixe de la chaîne 'ACGTAC' qui en soit aussi un suffixe ? Même question pour la chaîne 'TACTAC'.
3. La fonction suivante doit permettre, pour tout entier $i < len(m)$, de trouver le plus long suffixe de $m[:i]$ qui soit aussi un préfixe non initial de m.

```
def fonctionannexe(m) :
    L=[0]
    i=1
    j=0
    while i<n :
        if m[i]=m[j] :
            L.append(j+1)
            i=i+1
            j=j+1
        else
            if j>0 :
                j=L[j-1]
            else :
                L.append(0)
                i=i+1
            return L
```

   (a) Quel est le type de données renvoyé par cette fonction ?

   (b) Des erreurs de syntaxe (ou des oublis) se sont glissées dans cette fonction, corrigez-les (on ne demande pas de réécrire intégralement la fonction, mais simplement de signaler les changements à effectuer).

   (c) Décrire l'exécution de la fonction sur la chaîne de caractère 'ACAACA' (on donnera la valeur de chacune des variables à l'issue de chaque passage dans la boucle).

4. Le programme suivant effectue l'algorithme de recherche du motif :

```
def KMP(m,s) :
    L=fonctionannexe(s)
    i,j=0,0
    while i < len(s) :
        if s[i]==m[j] :
            if j==len(m)- 1 :
                return(i-j)
            else :
                i,j=i+1,j+1
        else :
            if j > 0 :
                i=L[j-1]
            else :
                i=i+1
    return -1
```

(a) Quel est le rôle des variables i et j dans ce programme ?

(b) Expliquer les lignes 6 et 7 du programme.

(c) Quel est le rôle de la liste L calculée à l'aide du programme fonctionannexe ?

IV. Recherche d'un motif en utilisant une structure de liste.

Une méthode beaucoup plus brutale pour rechercher un motif dans une chaîne de caractères consiste à procéder ainsi :

- on crée la liste de **toutes** les sous-chaînes de la chaîne complète.
- on trie cette liste par ordre alphabétique.
- on cherche si le motif recherché est présent dans cette liste alphabétique.

1. Donner la liste de toutes les sous-chaînes de la chaîne 'ACTCG'.
2. Si la chaîne complète compte n caractères, combien aura-t-elle au maximum de sous-chaînes distinctes ?
3. Écrire une fonction **indicemin(L)** qui prend comme argument une liste L de chaînes de caractères, et qui renvoie l'indice de la première chaîne de la liste par ordre alphabétique. On rappelle que la comparaison de chaînes de caractères à l'aide de $<$ effectue une comparaison alphabétique.
4. On écrit le programme suivant :

```
def mystere(L) :
    if len(L)==1 :
        return L
    i=indicemin(L)
    L1=L[ :i]+L[i+1 :]
    return L[i]+mystere(L1)
```

(a) Qu'effectue l'opération + sur les chaînes de caractère ?

(b) De quel type de programme s'agit-il ?

(c) Que fait ce programme ?

5. Quelle serait la meilleure méthode pour rechercher la présence du motif au sein de la liste triée de toutes les sous-chaînes (on ne demande pas d'écrire le programme correspondant) ?

V. Exploitation d'un base de données.

Un laboratoire spécialisé dans le séquençage d'ADN de bactéries dispose d'une base de données relationnelle constituée de deux tables :

- une table **ADN** contenant comme attributs un numéro d'identifiant nadn, le genre et l'espèce de la bactérie dont on a extrait l'ADN (une même bactérie n'apparait qu'une fois dans la liste).
- une table **sequencage** contenant comme attributs un numéro d'identification nseq, la date où le séquençage a été effctué, le nom du biologiste qui a effectué le séquençage, ainsi que le nadn de l'ADN séquencé et le nom du gène séquencé (lors d'un séquençage on ne décode qu'une petite partie de l'ADN complet de la bactérie, un même nadn peut donc intervenir plusieurs fois dans cette table).

Ci-dessous, un exemple de ce à quoi peuvent ressembler les premières lignes des deux tables :

nadn	**genre**	**espece**
2	Pseudomonas	syringae
8	Listeria	monocytogenes
12	Bacillus	cereus

nseq	**date**	**employe**	**nadn**	**gene**
1	28-04-2020	Martin	8	gyrB
2	18-12-2019	Dupont	8	leuS
3	15-09-2020	Martin	15	recA

1. Expliquer ce que va afficher la requête SQL suivante : SELECT Count(*) FROM sequencage WHERE date='05-06-2012'
2. Écrire la requête SQL correspondant à l'opération suivante de l'algèbre relationnelle : $\Pi_{nadn}(\sigma_{gene='leuS'}(sequencage))$
3. Écrire une requête SQL permettant d'afficher la liste des employés dont le nom commence par un 'G'.
4. Écrire une commande SQL permettant d'afficher tous les gènes séquencés sur les bactéries de genre « Bacillus ».
5. Écrire une requête SQL permettant d'afficher la liste des bactéries (genre et espèces) séquencées par M.Dupont le 10 mars 2018.