

PROGRAMMATION DYNAMIQUE 3/3

ALIGNEMENT DE SEQUENCES

- La molécule d'ADN, présente chez tous les êtres vivants, est composée de l'assemblage dans un ordre précis de quatre nucléotides :
 - A (adénine), T (thymine), G (guanine) et C (cytosine).

- Une séquence ADN est la succession des nucléotides qui le constituent.

- En bio-informatique, l'alignement de séquences (ou alignement séquentiel) est une manière de représenter deux ou plusieurs séquences d'ADN les unes sous les autres, de manière à en faire ressortir les régions homologues ou similaires. Ces alignements sont ainsi effectués par des programmes informatiques qui reposent sur des algorithmes très efficaces, basée sur la programmation dynamique.

- L'alignement a d'importantes utilisations en bio-informatique car il permet un certain nombre de prédictions, ou encore d'identifier des caractères d'un ancêtre commun à plusieurs espèces.



1. Présentation du problème

Considérons deux séquences ADN :

```
X = "ATTGCAT"  
Y = "AGTCCAG"
```

Un *alignement* de X et Y est une façon de les faire correspondre, en conservant l'ordre des caractères, mais en introduisant éventuellement des *trous* (représenté par le symbole " - "). En introduisant quelques trous dans X et dans Y , nous pouvons les aligner ainsi:

```
AT-TG-CA-T  
A-GT-CCAG-
```

Le *score* d'un alignement est le nombre de trous nécessaires à l'alignement. Pour l'exemple précédent, $\text{score}(X,Y)$ a pour valeur 6. On cherche à minimiser ce score.

Exercice 1 :

Aligner les mots "RESSORT" et "ESPRIT". Quel score minimal obtient-on ?

Réponse :

-
-
-
-

Remarque:

- Plusieurs solutions sont parfois possibles, donnant un même score minimum.

2. Une approche récursive

Exercice 2 :

On note `aligner_rec(X,Y,i,j)` la fonction qui prend en paramètres:

- les séquences `X` et `Y` ;
- un entier `i` compris entre 0 et $\text{len}(X)$;
- un entier `j` compris entre 0 et $\text{len}(Y)$.

La fonction `aligner_rec` doit retourner la valeur minimum du score si on se restreint aux `i` premiers caractères de `X` et aux `j` premiers caractères de `Y` .

1. Quelques exemples : Avec `X="GENOME"` et `Y="ENORME"`, écrire dans chaque cas ce que renvoie les appels ci-dessous et donner un alignement possible:

- `aligner_rec(X,Y,0,2)` :

- `aligner_rec(X,Y,3,0)` :

- `aligner_rec(X,Y,1,1)` :

- `aligner_rec(X,Y,2,1)` :

2. Cas de base : Que renvoie `aligner_rec(X,Y,i,j)` lorsque `i = 0` ou lorsque `j = 0` ?

Réponse :

3. Cas récursif : On suppose que `i` et `j` sont strictement supérieurs à zéro. Pour aligner les `i` premiers caractères de `X` et les `j` premiers caractères de `Y`, on compare le `i`-ème caractère de `X` et le `j`-ème caractère de `Y`, c'est à dire `X[i-1]` et `Y[j-1]` :

- S'ils sont identiques, alors le score est le même que celui de l'alignement des `i-1` premiers caractères de `X` et les `j-1` premiers caractères de `Y`, que l'on peut renvoyer récursivement.
 - Exemple : Avec `X="GENOME"` et `Y="ENORME"`, `aligner_rec(X,Y,3,2)` a le même score que `aligner_rec(X,Y,2,1)` car le 3ème caractère de `X` et le second caractère de `Y` sont identiques ('N').
- Sinon, il va falloir ajouter un trou, le score va nécessairement augmenter de 1, des appels récursifs à `aligner_rec` nécessaires, mais avec des paramètres différents pour `i` et pour `j` lors de ces appels.

- Exemple : Avec `X="GENOME"` et `Y="ENORME"`,
 - `aligner_rec(X,Y,3,4)` renvoie 3

```
GEN--
-ENOR
```

- `aligner_rec(X,Y,4,3)` renvoie 1

```
GENO
-ENO
```

- Pour obtenir `aligner_rec(X,Y,4,4)` , on renvoie le plus petit score des deux cas précédents auquel on ajoute 1, ce qui donne ici 1+1=2.

```
GENO-
-ENOR
```

A l'aide de ces informations et de celles de la question 2., compléter la fonction ci-dessous :

```
In [1]: def aligner_rec(x,y,i,j):
        '''renvoie le score(nbre de trous minimal)
        de x et y'''
        if i==0 : #si la chaîne x est vide
            pass #on renvoie la longueur de y (on a que des trous)
        elif j==0:
            pass
        else:
            #on s'intéresse au dernier caractère
            if x[i-1]==y[j-1]:
                pass
            else:
                pass
```

```
In [48]: #tests
assert(aligner_rec("ATTGCAT", "AGTCCAG",7,7)==6)
assert(aligner_rec("GENOME", "ENORME",6,6)==2)
assert(aligner_rec("RESSORT", "ESPRIT",7,6)==5)
```

Exercice 3 :

Tester la fonction précédente avec les séquences suivantes :

```
In [ ]: X = "TTCACCAGAAAAGAACACGGTAGTTACGAGTCCAATATTGTTAAACCG"
        Y = "TTCACGAAAAAGTAACGGGCCGATCTCCAATAAGTGCGACCGAG"

        aligner_rec(X,Y,len(X),len(Y))
```

Remarques :

- La version récursive calcule une solution correcte au problème, mais elle souffre de lenteur car un même sous-problème peut être plusieurs fois résolu. Cela se traduit par de multiples appels à `aligner_rec` avec des paramètres identiques.

3. Une approche dynamique

- Afin de ne pas recalculer plusieurs fois la solution à un même sous-problème, on propose d'utiliser et de compléter un tableau `score` à deux dimensions, où `score[i][j]` (pour $0 \leq i \leq \text{len}(X)$ et $0 \leq j \leq \text{len}(Y)$) stocke la valeur minimum du score qu'il est possible d'obtenir si l'on se restreint aux i premiers caractères de `X` et aux j premiers caractères de `Y`.
- Exemples : avec `X="GENOME"` et `Y="ENORME"`,
 - `score[4][3]` est le score minimal de l'alignement de "GENO" et "ENO"
 - `score[0][2]` est celui de la chaîne vide avec "EN"

Exercice 4 :

1. Quelle valeur devrait être stockée dans `score[i][j]` lorsque $i = 0$ ou lorsque $j = 0$?

Réponse :

- `score[i][0]` :
- `score[0][j]` :

2. Supposons maintenant que $i > 0$ et $j > 0$. En se basant sur le principe de la fonction récursive précédente, établir une formule (de récurrence) qui donne la valeur de `score[i][j]` en fonction de `score[i-1][j-1]` ou `score[i][j-1]` ou encore `score[i-1][j]` et compléter la fonction ci-dessous.

```
In [2]: def aligner_dyn(x,y):
        n=len(x)+1
        m=len(y)+1
        score=[[0 for _ in range(m)] for _ in range(n)] #tableau des scores

        #score lorsque j=0
        for i in range(n):
            pass

        #score lorsque i=0
        for j in range(m):
            pass

        #score[i][j] sinon
        for i in range(1,n):
            for j in range(1,m):
                pass

        return score[n-1][m-1]
```

```
In [12]: #test
X = "TTCACCAGAAAAGAACACGGTAGTTACGAGTCCAATATTGTTAAACCG"
Y = "TTCACGAAAAAGTAACGGGCCGATCTCCAATAAGTGCGACCGAG"

aligner_dyn(X,Y)
```

Out[12]: 26

Remarques :

- On ne peut que constater la différence de vitesse d'exécution sur des longues séquences par rapport à la version précédente !
- La complexité en temps de cet algorithme est en $O(n \times m)$.
- Il y a un coût en mémoire puisqu'il faut prévoir l'espace mémoire pour stocker le tableau.

4. Résolution du problème

- Afficher le score, c'est bien mais ce que l'on souhaite, c'est aussi afficher un alignement des séquences concernées. Pour cela, il faut modifier la version précédente, en particulier la variable `score` qui reste un tableau à deux dimensions mais qui contiendra désormais des tuples de trois éléments : le score, des caractères de X avec les trous, des caractères de Y avec des trous.
- Exemples : avec X="GENOME" et Y="ENORME",
 - `score[4][3]` contiendra (1, 'GEN0', '-ENO')
 - `score[0][2]` contiendra (2, '--', 'EN')

Exercice 5 :

En remplaçant les pointillées, compléter la fonction `aligner_dyn_affiche(x,y)` pour qu'elle renvoie un tuple contenant le score minimal de l'alignement de X et Y, l'alignement des caractères de X, l'alignement des caractères de Y.

```
In [7]: def aligner_dyn_affiche(x,y):
n=len(x)+1
m=len(y)+1
score=[[ (0, '', '') for _ in range(m)] for _ in range(n)] #tableau des scores et des motifs obtenus

for i in range(n):
    score[i][0]=(i,x[:i], '-'*i)

for j in range(m):
    score[0][j]=(j, '-'*j,y[:j])

for i in range(1,n):
    for j in range(1,m):
        if ...:
            total,xmotif,ymotif=score[i-1][j-1]
            score[i][j]=(total,...,...)
        else:
            total1,xmotif1,ymotif1=score[i][j-1]
            total2,xmotif2,ymotif2=score[i-1][j]

            if total1 <= total2:
                total=...
                xmotif=...
                ymotif=...
            else:
                total=...
                xmotif=...
                ymotif=...

            score[i][j]=(total,xmotif,ymotif)

    return score[n-1][m-1]
```

```
In [9]: #X = "ATTGCAT"
#Y = "AGTCCAG"

#X = "GENOME"
#Y = "ENORME"

#X = "RESSORT"
#Y = "ESPRIT"

X = "TTCACCAGAAAAGAACACGGTAGTTACGAGTCCAATATTGTTAAACCG"
Y = "TTCACGAAAAAGTAACGGGCCGATCTCCAATAAGTGCGACCGAG"
```

```
In [10]: score,x,y=aligner_dyn_affiche(X,Y)
print(x)
print(y)
print(score)

TTCACCAGAAAAGA--ACACGGTAGTTA-CGAG--TCCAATATT-GTTAA---ACC--G
TTCA-C-GAAAA-AGTA-ACGG--G---CCGA-TCTCCAATA--AG-T--GCGACCGAG
26
```