Exercice : Détermination de la plus longue sous-chaîne commune à deux chaines de caractères

Eléments de correction

En génétique par exemple, il peut s'avérer très important de savoir quels sont les points communs de deux génomes. Le problème de la plus longue sous-chaîne permet d'apporter une réponse à cette question.
On appelle **sous-chaine** d'une chaîne de caractères a = a1...an toute chaine de caractères s extraite de a telle que s = a_i ...a_k où i ≤ … ≤ k < n et ∀p ∈ {i , . . . , k}, a_p ∈ a. Les caractères de s n'apparaissent pas nécessairement de manière consécutive dans la chaîne a.
Soient a = a_1 . . . a_q et B = b_1 … b_p 2 chaines de caractères non vides. On **appelle plus longue sous-chaine commune** à a et b toute sous-chaîne commune à a et b de longueur maximale.
Si l'une des chaînes a ou b est vide ou si a et b n'ont aucune sous-chaîne commune, la chaine vide est alors l'unique plus longue sous-chaîne commune à a et b.
Exemple : les chaines de caractères "AAA" et "TAA" sont les plus longues sous-chaînes communes aux chaines de caractères "ATAGA" et "TAACA".

1. Pour les chaines a="AATGCG" et b="TATTAGC" ? Donner les solutions des plus longues suites communes.

Solution : ATGC et AAGC.

2. Écrire une **fonction de signature** **est_sousChaine (ch : str , sch : str)->bool** dont les ch1 et sch paramètres sont deux chaînes de caractères. Cette fonction renvoie True si sch est une sous-chaîne de ch, False sinon.

Solution :

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

3. Écrire une **fonction de signature** **est_commune(ch1 : str , ch2 : str, sch : str) -> bool** où les paramètres sont des chaînes de caractères. Cette fonction renvoie True si sch est une sous-chaîne commune à a et b.

Solution :

```
def est_commune(ch1, ch2, sch):
    return est_sousChaine(ch1, sch) and est_sousChaine(ch2, sch)
```

4. Le problème de la plus longue sous-chaîne commune entre a et b est noté L(a, b), son résultat est la longueur maximale d'une sous-chaîne commune à a et b.

Formuler le **problème L(a, b) récursivement** afin de pourvoir justifier de sa sous-structure optimale.

Solution :

Expression récursive - Sous-structure optimale

On se donne une chaîne a, de longueur n et on note a1, ... , an ses caractères.
De même on se donne une seconde chaîne b de longueur m et b1, ... , bm ses caractères.
On a alors 2 situations :
• le dernier caractère est commun : c = an = bm. Il est alors le dernier caractère de la plus longue sous-séquence commune et cette sous-séquence commune est obtenue en concaténant la plus longue sous-séquence commune de a1, ... , an−1 et b1, ... , bm−1 et c.
• le dernier caractère est différent : l'un au moins de xn et ym n'est pas dans cette plus longue sous-séquence commune.

			b_m
		(n-1,m-1)	(n-1,m)
a_n		(n,m-1)	(n,m)

On peut le retirer. La plus longue sous-séquence commune est la plus grande entre :
→ la plus longue sous-séquence commune entre a1, ... , a_{n-1} et b1, ... , b_m,
→ la plus longue sous-séquence commune entre a1, ... , a_n et b1, ... , b_{m-1},
On est donc ramené à chercher des sous-séquences maximales entre des sous-chaînes de a et b : on a bien l'apparition de sous-structure optimale dans la solution.

Une approche récursive directe va de nouveau amener à un problème de complexité exponentielle.

Chevauchement des sous-problèmes
Le calcul de la plus longue sous-séquence commune d'une portions de chacune des deux chaînes est utilisé par beaucoup d'autres calculs de portions plus grande. On a donc bien un chevauchement très important des sous-problèmes, ce qui invite de nouveau à un algorithme de programmation dynamique.

Soit L(i , j) la longueur de la plus longue sous-chaîne des chaînes extraites $a_1a_2 \ldots a_i$ et $b_1b_2 \ldots b_j$ des chaînes a et b.

On regarde si l'une des 2 chaines est vide.
Si non, on regarde si le dernier élément de chaque chaine est le même ou pas

On obtient la récurrence :
Pour tout i €[0,n] et pour tout j €[0,m],

- Si i = 0 ou j = 0, L(i,j) = 0
- Si ai≠bj L(i,j) = max (L(i-1,j) , L(i,j-1))
- Si ai=bj L(i,j) = 1 + L(i-1,j-1)