Recherche textuelle – algorithmes naïfs

Objectif : Ecrire une fonction recherche(motif,texte) qui prend en paramètre deux chaînes de caractère : motif et texte et qui retourne la liste des indices d'occurences du motif dans le texte.

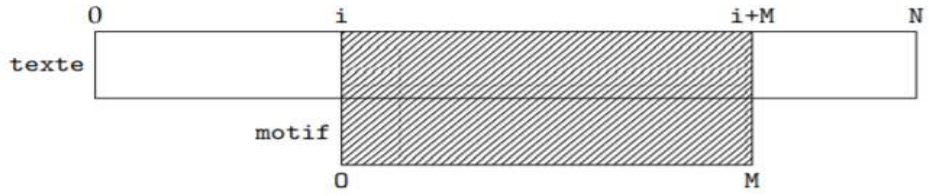
Par exemple :
recherche('ACAAGCGCACAAGACGCGGCAGACCTTTCGAACCTACTAGTGGGAGCACGGCCAATTCTCCTAGGCAAGTAGTGGTGAGCG','CAAG') retournera [1,9,65]

Pour cela faisons quelques rappels sur les chaines de caractères :

- Le caractère à l'indice i est obtenu par texte[i], il correspond à la lettre à la $(i + 1)^{ème}$ position dans texte ;

- La sous-chaîne qui contient les caractères d'indice de i inclus à j exclus est obtenue par texte[i:j]

Avec les conventions suivantes : N = len(texte) et M = len(motif), on obtient la représentation suivante :



Exercice 1 On donne : — le texte : AGAGGTCTGGCGCTTTGGAGT
— le motif : GAG

1. Combien d'occurrences du motif trouve-t-on dans le texte ?
2. Quelles sont leurs positions ?
3. Décrire l'algorithme utilisé, en langage naturel.

……
……
……
……

Exercice 2

1. Complétez le code de la fonction suivante :

```
def recherche_naive_v1(texte : str, motif : str):
    N,M = len(texte),len(motif)
    reponse = []
    for i in range(……..):
        if .....
            reponse.append(i)
    return reponse
```

Cette fonction prend en arguments deux chaines de caractères texte et motif et retourne une liste contenant les indices d'occurrence du motif dans le texte. S'il n'y aucune occurrence alors une liste vide est retournée

2. Écrire cette fonction et ajouter 5 tests pour vérifier le bon fonctionnement de cette fonction.

Il faut savoir que l'extraction d'une sous chaine de caractère avec l'opération chaine[i:j] est coûteuse. Ce coût est proportionnel aux nombres de caractères de la sous chaine extraite.

Estimez le coût des extractions des sous chaine pour l'algorithme précédente : . . . . . .

On peut essayer d'améliorer la situation.

Une première idée est de ne plus utiliser l'extraction de sous chaine de caractère. On va remplacer cette méthode par une boucle qui va comparer le motif avec le texte caractères par caractères. L'indice de recherche va s'incrémenter dès que deux caractères ne coïncident pas. On économise ainsi du temps de calcul.

Exercice 3 : Écrire une fonction recherche_naive_v2 basée sur le principe vu ci-dessus.

La structure conditionnelle devra être remplacée par une boucle pour tester la correspondance entre le motif et le texte caractères par caractères.