Recherche textuelle

1.Exemples

Dans de nombreux domaines, nous sommes amenés à rechercher une sous chaine de caractères dans une chaine de caractères plus grande.

1.1 La recherche d'un mot dans un texte littéraire :

> "Aragorn s'agenouilla alors auprès de Faramir et posa une main sur son front ; et les observateurs sentirent qu'une formidable lutte était en train de se jouer. Car le visage d'Aragorn devint gris de fatigue ; et de loin en loin, il appelait le nom de Faramir, mais ses appels se faisaient toujours plus faibles à leur ouïe, comme si Aragorn lui-même s'éloignait d'eux, marchant dans quelque vallée lointaine et ténébreuse, appelant une âme égarée. Il prit alors deux feuilles, qu'il déposa dans ses mains, puis il souffla dessus et les écrasa ; et d'emblée, une fraîcheur vivifiante embauma toute la pièce, comme si l'air même s'éveillait et picotait, pétillant de joie. Alors, il jeta les feuilles dans les bols d'eau fumante qu'on lui avait apportés, et tous les cœurs aussitôt s'apaisèrent. Car le parfum qui vint à chacun était comme le souvenir de matins humides de rosée et gorgés de soleil, dans un pays où la beauté printanière du monde n'est elle-même qu'un souvenir fugitif. Mais Aragorn se leva comme revigoré, et ses yeux souriaient tandis qu'il tenait l'un des bols devant le visage de Faramir, tout enveloppé de rêves."

(J.R.R Tolkien "Le Seigneur des Anneaux" traduction par Daniel Lauzon).

Q1 : Utiliser un traitement de textes (ctrl + F) pour rechercher combien de fois le nom "Aragorn" est cité.

4 fois.

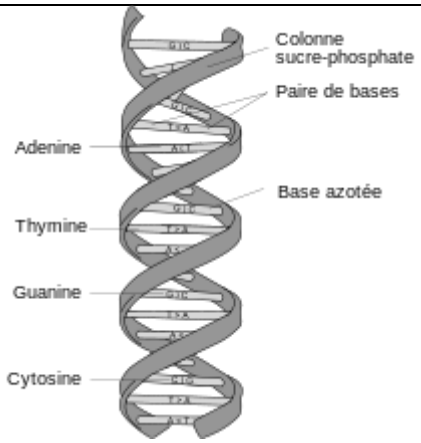
1.2 En biologie : rechercher un motif dans une séquence d'ADN

On rappelle que l'information génétique présente dans nos cellules est portée par les molécules d'ADN.

Les quatre bases nucléiques constitutives de l'ADN sont l'adénine (A), la cytosine (C), la guanine (G) et la thymine (T)

Voici un exemple de séquence :

"GGCAGCCGAACCGCAGCAGCAC"



Source : Wikipédia

Q2 : dans cette séquence combien de fois a-t-on le motif "GCAG" ? :

3 fois

Q3 : en quelles positions (on note 0 l'indice de la première lettre)

1, 12 et 15. GGCAGCCGAACCGCAGCAGCAC

2.Vocabulaire, notations pour toute cette activité

Texte = suite de caractères, pris dans un alphabet donné.

Motif : c'est aussi un texte, mais de « petite » taille.

La taille du motif est obligatoirement **inférieure** à la taille du texte.

Dans un texte littéraire l'alphabet est par exemple : Alphabet = {A,B...Z,a,b...Z,0,1...9, caractères spéciaux}

Rechercher un motif M dans un texte T signifie :

- Rechercher de toutes les occurrences de M dans T
- Effectuer une recherche exacte ⇒ on n'autorise pas d'erreur !

Q4 : Quel est l'alphabet d'une séquence d'ADN ?

{A, G, C, T}

3.Algorithme naïf : recherche par fenêtre glissante

3.1 première version

Cette version utilise l'outil de manipulation des chaines de caractères vu en 1ere.

Exemple :

txt='bonjour maman' print(txt[3:10])	**Résultat dans la console :** jour ma

```
def recherche_fenetre_V1(texte,motif):
    longTexte = len(texte)
    longMotif=len(motif)
    for i in range(longTexte-longMotif+1):
        #extraction
        extraction=texte[i:i+longMotif]
        print(extraction)
        #test
        if extraction==motif:
            print("trouvé en position",i)



recherche_fenetre_V1('bonjour papa maman',
'papa')
```

Résultat dans la console :

```
bonj
onjo
njou
jour
our
ur p
r pa
 pap
papa
trouvé en position 8
apa
pa m
a ma
 mam
mama
aman
```

Dès que le mot « papa » est trouvé, le programme affiche sa position (indice du 1er caractère) dans le texte.

3.2 Deuxième version

Ce programme sort de la boucle dès que le 1er mot recherché est trouvé

```
def recherche_fenetre_V2(texte, motif):
    longTexte = len(texte)
    longMotif=len(motif)
    for i in range(longTexte-longMotif+1):
        recherche = True
        k=0
        while recherche and k < longMotif:
            print(texte[i+k],end="")
            if motif[k] != texte[i+k]:
                recherche = False
            k += 1
        print()
        if recherche:
            return True
    return False


print(recherche_fenetre_V2('bonjour papa maman',
'papa'))
```

Résultat dans la console :

```
b
o
n
j
o
u
r

papa
True
```

A chaque itération dans la boucle pour, on vérifie si la lettre du texte correspond à la 1ere lettre du mot recherché :

- La lettre correspond, on reste dans la boucle while
- La lettre ne correspond pas, on sort de la boucle while pour passer à la prochaine lettre du texte.

La complexité de cet algorithme est quadratique dans le pire des cas $O(N^2)$. Donc peu performant.

Nous allons voir qu'il est beaucoup plus efficace de faire la recherche à l'envers à partir de la fin du mot.

4.L'algorithme de Boyer-Moore

Cet algorithme de Boyer-Moore repose sur deux idées :

- On compare le mot de droite à gauche à partir de sa dernière lettre.
- On n'avance pas dans le texte caractère par caractère, mais on utilise un décalage dépendant de la dernière comparaison effectuée.

Nous considérons ici la recherche du motif = 'dab' dans le texte texte = 'abracadabra'.

On commence la recherche à l'index 2 :

```
abracadabra
dab
```

Il n'y a pas de correspondance à la fin du mot : 'r' != 'b', donc on avance, mais de combien de caractères avance-t-on ? Pour le décider, on utilise le fait que le caractère 'r' n'apparait pas dans le mot cherché, donc on peut avancer de n = len(mot) = 3 caractères sans crainte de rater le mot.

On recherche donc à l'indice 2 + 3 = 5 :

```
abracadabra
   dab
```

Il n'y a pas de correspondance à la fin du mot : 'a' != 'b', donc on avance, cependant, cette fois, comme le caractère 'a' apparait pas dans le mot cherché en avant-dernière position, on ne peut avancer que de une case pour faire une comparaison en alignant les 'a'.

On recherche donc à l'indice 5 + 1 = 6 :

```
abracadabra
    dab
```

Il n'y a pas de correspondance à la fin du mot : 'd' != 'b', donc on avance, cependant, cette fois, comme le caractère 'd' apparait dans le mot cherché en avant-avant-dernière position(*première position, mais on doit lire à l'envers !*), on avance de deux cases pour faire une comparaison en alignant les 'd'.

On recherche donc à l'indice 6 + 2 = 8 :

```
abracadabra
      dab
```

Maintenant lorsqu'on effectue les comparaisons à l'envers : les 'b', puis les 'a', puis les 'd' correspondent. On a trouvé le mot on renvoie VRAI.

La complexité d'un algorithme de recherche textuelle Boyer-Moore se mesure essentiellement par le nombre d'opérations de comparaison de caractères qu'il effectue. L'analyse précise de la complexité de l'algorithme que nous avons programmé est difficile, et dépasse le niveau attendu en enseignement NSI.

```
def pre_traitement(mot):
    #Renvoie un dictionnaire avec pour clé la lettre et pour valeur le décalage
    n = len(mot)
    decalages = {}
    # Il n'est pas nécéssaire d'inclure la dernière lettre
    for i, letter in enumerate(mot[:-1]):
        decalages[letter] = n - i -1
    return decalages

def recherche_mot_boyer(texte, motif):
    #Recherche un mot dans un texte avec l'algo de boyer-moore
    longTexte = len(texte)
    longMotif=len(motif)
    # création de notre dictionnaire de décalages
    decalages = pre_traitement(motif)
    print(decalages)
    # on commence à la fin du mot
    i = longMotif - 1
    while i < longTexte:
        lettre = texte[i]
        if lettre == motif[-1]:
            # On vérifie que le mot est là avec un slice sur texte
            # On pourrait faire un while
            if texte[i-longMotif+1:i+1] == motif:
                return True
        # on décale
        if lettre in decalages.keys():
            i += decalages[lettre]
        else:
            i += longMotif

    return False


print(recherche_mot_boyer('abracadabra', 'dab'))
```

5.Exercices

Appliquer l'algorithme de Boyer-Moore sur le texte et le motif suivant

texte="GGCCAGCCGAACCGCAGCAGCAC"
motif="GCAG"

```
print(recherche_mot_boyer('GGCCAGCCGAACCGCAGCAGCAC', 'GCAG'))
```

Modifier le programme afin d'afficher la position (indice) du motif dans le texte

```
if texte[i-longMotif+1:i+1] == motif:
                print("position",i-longMotif+1)
                return True
```

Faire une recherche internet sur les chercheurs **Robert S. Boyer** et **J Strother Moore**.