Algorithme de Boyer-Moore

Table des matières

1) Présentation et applications

Un des problèmes classiques en informatique est la recherche d'un motif dans une chaîne de caractères. Par exemple, en bio-informatique, il est parfois nécessaire de déterminer si une séquence de gènes est présente ou non dans un échantillon d'ADN, voire même de déterminer sa position dans cet échantillon. Il est donc important de disposer d'un algorithme efficace et qui demande peu de temps de calcul.

2) Recherche naïve

Pour trouver la position d'un motif dans une chaîne de caractères, on peut utiliser un algorithme "naïf", c'est-à-dire le premier auquel on pense.

```python
def recherche_naive(texte, motif):
    n = len(texte)  #Longueur du texte
    m = len(motif)  #Longueur du motif à rechercher
    indices = []    #Tableau des indices des débuts de motif trouvés dans le texte

    for i in range(n - m + 1):
        j=0 #Indice de la position actuelle dans le motif
        while j < m and texte[i + j] == motif[j]:
            j += 1
        if j == m:
            indices.append(i)

    return indices
```

On va donc s'arrêter sur chaque caractère du texte dans lequel on recherche le motif et on va vérifier, à chaque fois, si oui ou non le motif est présent (s'il commence à ce caractère). Cet algorithme est très coûteux en temps (complexité O(n*m) avec n la taille du texte et m la taille du motif).

3) Algorithme

L'exécution de l'algorithme de Boyer-Moore se décompose en 2 étapes :

1. Le prétraitement

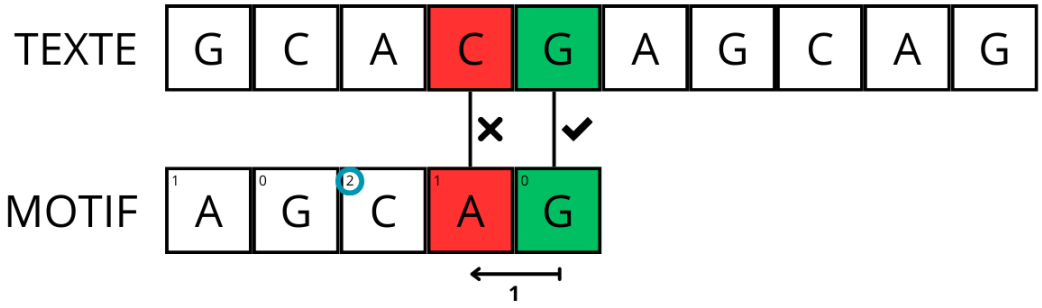
Lors de cette première étape, on va créer un tableau de décalages. À chaque lettre du motif recherché, on va associer sa plus courte distance avec le caractère à droite du motif. Par exemple, pour le motif "AGCAG", on aura le tableau suivant :

CARACTÈRE	POSITION	DÉCALAGE
A	3	1
G	4	0
C	2	2

(Les positions commencent à 0 à gauche du motif)

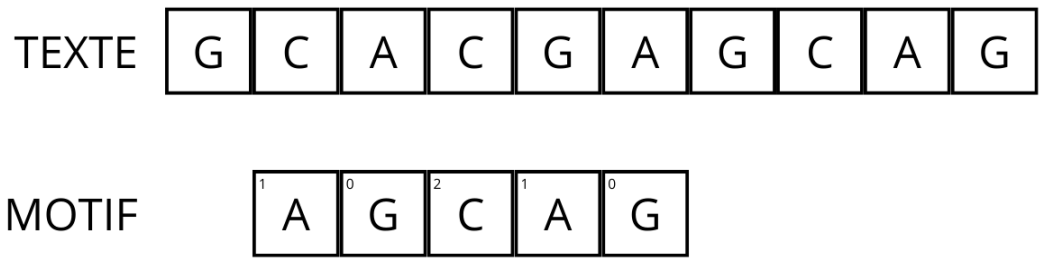
2. Les comparaisons / sauts

Cette étape se répète jusqu'à ce que le motif soit trouvé. Lors de cette étape, on va "positionner" le motif par rapport au texte, puis comparer les caractères du motif et du texte 1 par 1, en partant du caractère à droite du motif vers les caractères à gauche. À chaque comparaison, on vérifie si les caractères du texte et du motif (à la même position) sont semblables ou non. S'ils le sont, on continue la comparaison à gauche. Sinon, on utilise le tableau de décalages définit à l'étape 1 pour "décaler" le motif vers la droite. Par exemple, si la lettre du motif est un "A" et celle du texte un "C", on regarde à la ligne C du tableau et on "décale" le motif par rapport au texte vers la droite de 2 caractères (colonne "DÉCALAGE") moins la position du "A" depuis la fin du motif. Avec un schéma cela donne (on garde le même tableau de décalages) :



2 - 1 = 1

On décale le motif de 1 vers la droite :



Et on recommence jusqu'à trouver le motif dans le texte !

Vous pouvez visualiser l'exécution de cet algorithme ici ou en scannant ce QR-Code :

Voici une des implémentation l'algorithme de Boyer-Moore en Python :

```python
def boyerMoore(texte, motif):

    m = len(motif)  #Longueur du motif à rechercher
    n = len(texte)    #Longueur du texte dans lequel on cherche le motif

    tab_sauts = {motif[i]:i for i in range(m)}  #Tableau contenant les sauts correspondants aux caractères du motif

    decalage = 0    #Indice de la position actuelle dans le texte
    indices = []    #Tableau des indices des débuts de motif trouvés dans le texte
    i = m - 1   #Indice de la position actuelle dans le motif

    while decalage <= n - m:
        i = m - 1

        while i >= 0 and motif[i] == texte[decalage + i]:
            i -= 1

        if i < 0:
            indices.append(decalage)
            if decalage + m < n:
                decalage += m - tab_sauts.get(texte[decalage + m], -1)
            else:
                decalage += 1
        else:
            decalage += max(1, i - tab_sauts.get(texte[decalage + i], -1))

    return indices
```

Il n'est pas important de connaître par cœur cette implémentation, mais il est essentiel de réussir à la comprendre !