Numérique et Sciences Informatiques
Chapitre XIII - Recherche textuelle

Classe de terminale

Le but de ce chapitre est de rechercher un motif (une chaîne de caractères) dans un texte (une chaîne de caractère de longueur supérieure à celle du motif en général).

Nous prendrons l'exemple tiré de la bio-informatique où l'on recherche le motif « $ATATAC$ » dans le brin d'ADN suivant :

$$ATAGACACAATATACTGACACGAT$$

I. Recherche naïve

I.1. Principe

La première idée de recherche d'un motif dans un texte est la suivante :

- Je compare le 1er caractère du motif avec le 1er caractère du texte.
- Si les deux caractères sont égaux, je compare le deuxième caractère du motif avec le deuxième caractère du texte. Et ainsi de suite.
- Sinon je décale le motif d'un caractère et je réitère le procédé en commençant avec le 2ème caractère du texte.

Pour notre exemple, les différentes étapes donneraient :

$A\ T\ A\ G\ A\ C\ A\ C\ A\ A\ T\ A\ T\ A\ C\ T\ G\ A\ C\ A\ C\ G\ A\ T$
✓ ✓ ✓ ×
A T A T A C

$A\ T\ A\ G\ A\ C\ A\ C\ A\ A\ T\ A\ T\ A\ C\ T\ G\ A\ C\ A\ C\ G\ A\ T$
  ×
  $A\ T\ A\ T\ A\ C$

$A\ T\ A\ G\ A\ C\ A\ C\ A\ A\ T\ A\ T\ A\ C\ T\ G\ A\ C\ A\ C\ G\ A\ T$
    ✓ ×
    $A\ T\ A\ T\ A\ C$

$A\ T\ A\ G\ A\ C\ A\ C\ A\ A\ T\ A\ T\ A\ C\ T\ G\ A\ C\ A\ C\ G\ A\ T$
      ×
      $A\ T\ A\ T\ A\ C$

...

I.2. Implémentation de cet algorithme

```
1  def recherche(motif,texte,depart=0):
2      i=0
3      present=True
4      while i<len(motif) and present:
5          if motif[i]!=texte[depart+i]:
6              present=False
7          i+=1
8      if present:
9          return depart
10     else:
11         return recherche(motif,texte,depart+1)
```

On peut donc imaginer que dans le pire des cas, notre algorithme utilise environ $longueur(texte) \times longueur(motif)$ comparaisons.
On s'aperçoit aussi dans notre exemple que lorsqu'on décale le motif de 1 rang vers la droite, on est sûr que la comparaison suivante entre le motif et le texte sera fausse.
Il faudrait donc trouver un moyen de décaler le motif de plusieurs rangs et donc de trouver un bon décalage.

II. Algorithme de Boyer-Moore

L'algorithme de Boyer-Moore se fait en trois étapes :

- prétraitement du motif.
- comparaison du motif et de la chaîne en commençant par la droite du motif.
- trouver le bon décalage du motif pour la comparaison suivante.

II.1. Prétraitement du motif

Le prétraitement du motif M consiste en la création d'un tableau indiquant la dernière position de chaque symbole du motif dans le sous-motif (la sous-chaîne de caractères) $M[0..i]$ avec $0 \leq i < longueur(M)$.

Pour notre exemple, avec le motif $ATATAC$, on aurait :

i	A	T	C
0	0	-	-
1	0	1	-
2	2	1	-
3	2	3	-
4	4	3	-
5	4	3	5

II.2. Comparaison du motif et de la chaîne

On compare le motif et la chaîne en partant de la droite du motif.
Sur notre exemple :

$A\ T\ A\ G\ A\ C\ A\ C\ A\ A\ T\ A\ T\ A\ C\ T\ G\ A\ C\ A\ C\ G\ A\ T$
$\quad\quad \times\ \checkmark\ \checkmark$
$A\ T\ A\ T\ A\ C$

II.3. Trouver le bon décalage du motif

On trouve le bon décalage en regardant le prétraitement du motif.

$A\ T\ A\ G\ A\ C\ A\ C\ A\ A\ T\ A\ T\ A\ C\ T\ G\ A\ C\ A\ C\ G\ A\ T$
$\quad\quad \times\ \checkmark\ \checkmark$
$A\ T\ A\ T\ A\ C$

Dans notre exemple, l'erreur se fait sur un G du texte et sur le 4ème caractère du motif.
On regarde dans le prétraitement ou se trouve le G sur les 4 premiers caractères du motif (ligne du tableau où $i = 3$) : il n'y est pas, donc on décale le motif de 4 caractères.

Continuons les comparaisons :

$A\ T\ A\ G\ A\ C\ A\ C\ A\ A\ T\ A\ T\ A\ C\ T\ G\ A\ C\ A\ C\ G\ A\ T$
$\quad\quad\quad\quad\quad\quad\quad\quad \times$
$\quad\quad\quad\quad A\ T\ A\ T\ A\ C$

Ici, l'erreur se fait sur le A du texte et sur le dernier caractère (le 6ème) du motif.
Dans le prétraitement, on cherche où se trouve le dernier A dans les 6 derniers caractères du motif (ligne du tableau où $i = 5$. Il se trouve à la position n°4 soit le 5ème caractère. On décalera donc de $6 - 5 = 1$ caractères.

$A\ T\ A\ G\ A\ C\ A\ C\ A\ A\ T\ A\ T\ A\ C\ T\ G\ A\ C\ A\ C\ G\ A\ T$
$\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \times$
$\quad\quad\quad\quad\quad A\ T\ A\ T\ A\ C$

Ici, c'est le T du texte qui bloque sur le 6ème caractère du motif. Or le T se trouve en position n°3 du sous-motif composé de 6 caractères, soit le 4ème caractère. $6 - 4 = 2$ donc on décale de 2 le motif.

$$\begin{array}{c} A\ T\ A\ G\ A\ C\ A\ C\ A\ A\ T\ A\ T\ A\ C\ T\ G\ A\ C\ A\ C\ G\ A\ T \\ \times \\ A\ T\ A\ T\ A\ C \end{array}$$

Et ainsi de suite :

$$\begin{array}{c} A\ T\ A\ G\ A\ C\ A\ C\ A\ A\ T\ A\ T\ A\ C\ T\ G\ A\ C\ A\ C\ G\ A\ T \\ \checkmark\ \checkmark\ \checkmark\ \checkmark\ \checkmark\ \checkmark \\ A\ T\ A\ T\ A\ C \end{array}$$

En y réfléchissant bien, il serait plus judicieux d'écrire dans le prétraitement le décalage du motif à apporter plutôt que la position du symbole.

II.4. Implémentation

II.4.a. Implémentation du prétraitement

```python
def pretraitement(motif):
    resultat = []
    liste_symbole = []
    for symbole in motif:
        if symbole not in liste_symbole:
            liste_symbole.append(symbole)
    for i in range(1, len(motif) + 1):
        nouveau_motif = motif[0:i]
        resultat.append(dict())
        for symbole in liste_symbole:
            indice = len(nouveau_motif) - 1
            fait = False
            while indice >= 0 and not fait:
                if nouveau_motif[indice] == symbole:
                    fait = True
                else:
                    indice = indice - 1
            if indice == -1:
                resultat[i - 1][symbole] = i
            else:
                resultat[i - 1][symbole] = i - indice -1
    return resultat
```

II.4.b. Implémentation de l'algorithme de Boyer-Moore

```
def recherche_BOYER_Moore(motif, texte, indice_depart=0, decalage=None):
    if decalage == None:
        decalage = pretraitement(motif)
    if indice_depart > len(texte)-len(motif):
        return None
    presence = True
    i = len(motif) - 1
    while presence and i >= 0:
        if motif[i] != texte[indice_depart + i]:
            presence = False
        i = i - 1
    if i == -1:
        return indice_depart
    else:
        symbole = texte[indice_depart + i + 1]
        if symbole not in decalage[i + 1]:
            decal = i + 1
        else:
            decal = decalage[i + 1][symbole]
        return recherche_BOYER_Moore(motif, texte, indice_depart + decal, decalage)
```

Remarque

L'algorithme de Boyer-Moore a été implémenté ci-dessus en programmation dynamique avec un programme récursif en utilisant un paradigme fonctionnel.