Corrigé – Recherche textuelle : algorithme de Boyer-Moore

Exercice 1

1. La séquence s'arrête à l'indice n-1, mais puisqu'on cherche la correspondance du motif en entier, on peut s'arrêter à l'indice n-m-1.
2. j stocke le nombre de caractères qui coïncident entre S et M, à une position i donnée, en commençant par la gauche de M.
3. Deux cas sont possibles : à la fin de la boucle bornée, c'est-à-dire lorsqu'on a parcouru toute la séquence sans trouver le motif, ou bien lorsqu'on le trouve : on est dans ce cas lorsque j vaut m.

Exercice 2

1. On commence par comparer le dernier caractère du motif, puis l'avant-dernier, etc.
2. Lorsque qu'on arrive à un caractère dans la séquence qui ne coïncide pas avec caractère du motif à la même position.
3. Si l'élément de la séquence n'est pas dans le motif, on décale le motif à la position de l'élément de la séquence + 1.

   Si l'élément de la séquence considéré est à une autre place dans le motif, on décale le motif de manière à le faire coïncider avec ce caractère de la séquence, en prenant le caractère du motif le plus à droite possible, mais sans le faire revenir en arrière.

Exercice 3

```
----------------------
VOICI UN SIMPLE EXEMPLE
EXEMPLE
```
-> NON
U/E ko et U n'est pas dans le motif donc on place le motif après U

```
----------------------
VOICI UN SIMPLE EXEMPLE
       EXEMPLE
```
-> NON
L/E ko mais L dans motif donc on les aligne

```
----------------------
VOICI UN SIMPLE EXEMPLE
        EXEMPLE
```
-> NON
I/E ko et I pas dans le motif donc on place le motif après I

```
----------------------
VOICI UN SIMPLE EXEMPLE
           EXEMPLE
```
-> NON
E/X ko mais X dans le motif donc on les aligne

```
----------------------
VOICI UN SIMPLE EXEMPLE
                EXEMPLE
```
-> OUI !

Implémentation Python

1. (a) 0 1 3 4

   (b)
   ```python
   def calcule_table_derniere_occurrence(chaine):
       table_derniere_occurrence = {}
       for j in range(len(chaine)):
           table_derniere_occurrence[chaine[j]] = j
       return table_derniere_occurrence
   ```

   (c)
   ```python
   assert calcule\_table\_derniere\_occurrence("HELLO") == {"H": 0, "E": 1, "L": 3,
   ↪  "O": 4}
   ```

Test de la correspondance

1. Compléter le pseudo-code suivant.

```
fonction correspondance(sequence, motif, i, table_derniere_occurrence)
    Entrée:
        - sequence: séquence (chaîne de caractères)
        - motif: motif recherché (chaîne de caractères)
        - idx_sequence: position actuelle dans le sequence, correspondant à la
          position du caractère le plus à gauche du motif (entier)
        - table_derniere_occurrence: table de sauts (dictionnaire)

    Sortie:
        - decalage (entier) pour la prochaine étape, ou 0 si la
          correspondance a été trouvée

    Pour idx_motif variant de l'indice du dernier caractère du motif au premier de
    manière décroissante:
        lettre_sequence ← caractère à l'indice `idx_sequence + idx_motif` de sequence
        si lettre_sequence est différent de la lettre à l'indice idx_motif:
            si lettre_sequence n'est pas dans table_derniere_occurrence:
                renvoyer l'indice suivant la position courante
            renvoyer <la plus grande valeur entre l'indice de la dernière occurrence
            de la lettre qui coïncide dans le motif et 1 (car on ne veut pas "reculer")
            ex :
            FTGAABCF
             AXCA>

    renvoyer <renvoyer 0 (on a trouvé la correspondance)>
```

2. *Voir fichier Python*

Tests

Voir fichier Python