TD 3 – Recherche textuelle

Exercice 1. *Algorithme naïf*

✎ Programmer l'algorithme RechercheNaïve de telle sorte qu'il renvoie, en plus de la liste des positions de x dans t, le nombre de comparaisons effectuées.

Exercice 2. *Dernières occurences non finales*

1. Écrire un algorithme qui calcule un tableau associatif d tel que pour toute lettre a présente dans le motif, $d[a]$ est la position de la dernière occurrence non finale de a dans le motif.
2. L'algorithme simplifié de Boyer et Moore nécessite d'avoir également $d[b] = -1$ pour toutes les lettres b qui n'apparaissent pas dans le motif. On pourrait donc étendre le tableau d pour qu'il contienne toutes les lettres possibles. Quel est l'inconvénient de cette solution ?
3. Quel type de structure de données peut-on utiliser en Python pour pallier cette difficulté ?
4. Programmer l'algorithme BoyerMoore-MC en comptant les comparaisons effectuées.

Exercice 3. *Préfixes et suffixes*

1. Écrire un algorithme qui étant donné j, teste si $x_{[0,m-j[} = x_{[j,m[}$.
2. En déduire un algorithme de complexité $O(m^2)$ qui prend en entrée le motif x et renvoie un tableau p tel que $p[j] = p_x(j)$.
3. Écrire un algorithme qui étant donné j, calcule la longueur ℓ du plus long suffixe de x tel que $x_{[m-\ell,m[} = x_{[j-\ell,j[}$.
4. En déduire un algorithme de complexité $O(m^2)$ qui prend en entrée le motif x et renvoie un tableau s tel que $s[j] = s_x(j)$.
5. Programmer l'algorithme BoyerMoore-BS en comptant les comparaisons effectuées.

Exercice 4. *Algorithme général*

1. Programmer l'algorithme BoyerMoore en comptant les comparaisons effectuées.
2. Tester vos algorithmes sur différents jeux de données, et comparer le nombre de comparaisons effectuées.
   Vous pouvez utiliser les fichiers du dossier `chr18` : le fichier `chr18_0` contient l'ADN du chromosome 18 chez un être humain [1]*, et les fichiers suivants sont des sous-séquences du même ADN. On doit donc retrouver le motif `chr18_i` dans `chr18_j` si $i \geq j$. Pour récupérer le contenu d'un fichier comme chaîne de caractère, on peut utiliser le code suivant.*

```
with open("chr18/chr18_2") as c:
    s = c.read().replace('\n','')
```

1. Si j'ai bien compris…