Aix*Marseille université
Initiative d'excellence
Année 2019-20

Diplôme Inter-Universitaire
Enseigner l'informatique au lycée

Bloc 5 : TP 7

Recherche textuelle

1 Algorithme naïf de recherche de motif

Question 1 Implémenter l'algorithme de recherche naïve d'un motif dans un texte : la fonction devra s'arrêter dès qu'elle a trouvé une occurrence du motif dans le texte et renvoyer sa position. Quelle est sa complexité dans le pire des cas ? et dans le meilleur des cas ?

Question 2 Comment modifier la fonction pour qu'elle renvoie la liste de toutes les positions où apparaissent le motif dans le texte ?

2 Recherche d'un motif à l'aide d'un automate fini

Question 3 Construire l'automate de recherche du motif $M = aabab$ et illustrer son action sur le texte $T = aaababaabaababaab$.

Question 4 Construire l'automate de recherche du motif $M = bab$ et illustrer son action sur le texte $T = aabbababaababa$.

3 Algorithme de Boyer-Moore-Horspool

L'algorithme de Boyer-Moore-Horspool consiste à : - comparer le motif M avec un facteur $T[p..p+m-1]$ du texte, en partant de la droite du motif ; - et si une différence entre M et ce facteur est constatée (ou si on a trouvé une occurrence du motif mais qu'on les veut toutes), on décale le motif vers la droite de manière à faire coïncider la dernière lettre du facteur, c'est-à-dire la lettre $T[p+m-1]$, avec son occurrence la plus à droite dans M.

Question 5 Exécuter l'algorithme de Boyer-Moore-Horspool pour la recherche du motif `aababab` dans le texte `aabbbababacaabbaba`. Même question pour la recherche du motif `ete` dans le texte `la bete a ete etetee`. Compter le nombre de comparaisons effectuées et comparer ce nombre à celui nécessaire dans la recherche naïve.

Question 6 Trouver un exemple où le nombre de comparaisons nécessaires dans l'algorithme de Boyer-Moore-Horspool est *de l'ordre de* nm comparaisons, pour n et m quelconques (avec $m \leq n$).

Question 7 Implémenter une fonction `derniere_occurrence(motif, alphabet)` qui prend le motif en argument, ainsi qu'un ensemble de lettres (qui doit au moins contenir toutes les lettres du motif) et retourne un dictionnaire associant à chaque lettre le décalage à effectuer dans l'algorithme de Boyer-Moore-Horspool s'il s'agit de la dernière lettre du facteur. En particulier, on doit avoir `derniere_occurrence("aababab", {'a', 'b', 'c'}) = {'a': 1, 'b': 2, 'c': 7}`.

Question 8 L'utiliser pour implémenter l'algorithme de Boyer-Moore-Horspool consistant à utiliser l'algorithme naïf mais à utiliser un décalage permettant d'aligner la lettre courante du texte avec sa dernière occurrence dans le motif. On renverra à nouveau une liste de toutes les occurrences des positions où apparaissent le motif dans le texte.

Question 9 L'étude de complexité de l'algorithme de Boyer-Moore-Horspool est difficile. On peut cependant l'étudier par expérimentation. Télécharger le premier tome de l'ouvrage *Les misérables* de Victor Hugo sur le cours Ametice. L'importer dans Python à l'aide du code

```
with open('miserables-tome1.txt') as fh:
    texte = fh.read()
```

Pour tester le temps d'exécution, on utilisera le code suivant (la répétition de 5 appels permet de limiter les erreurs de mesure dues à l'ordonnanceur du système d'exploitation...) :

```
# Pour tester le temps d'exécution:
from timeit import timeit

def test():
    recherche(texte, 'Valjean')
# Afficher en secondes le temps d'exécution de 5 appels à test
print(timeit('test()', 'from __main__ import test', number = 5))
```

Comparer alors les temps d'exécution de l'algorithme naïf et de l'algorithme de Boyer-Moore-Horspool : vous devriez trouver un temps de calcul entre 2 et 3 fois inférieur via l'algorithme de Boyer-Moore-Horspool. Augmenter la taille du motif (en recherchant par exemple la phrase `'vous êtes le forçat Jean Valjean'`) pour observer l'évolution du temps de calcul des deux algorithmes : attention au fait que le temps de calcul n'est pas une mesure correcte du nombre de comparaisons effectuées...

4 Algorithme de Boyer-Moore (largement emprunté au chapitre 10 de l'ouvrage *Éléments d'algorithmique* de Beauquier, Berstel, Chrétienne)

Il est vraisemblable que le programme de Terminale NSI, qui mentionne l'algorithme de Boyer-Moore, pense sans doute plutôt à l'algorithme de Boyer-Moore-Horspool développé avant. Cependant, il faut savoir que l'algorithme original de Boyer-Moore est un peu différent, même s'il suit la même idée... Dans la version d'Horspool, le décalage du motif est déterminé par la dernière lettre du facteur examiné. À la place, l'algorithme de Boyer-Moore utilise la lettre du facteur où la différence a été constatée. Ainsi, la superposition du motif $x = aababab$ au facteur $aabcbab$ conduit, dans l'algorithme de Horspool, à un décalage de 2 à cause de la lettre b terminant le facteur, et ceci indépendamment du fait que c'est à l'occurrence de la lettre c dans le facteur que la différence a été constatée. En l'occurrence, la lettre c n'apparaissant pas dans le motif, le décalage proposé peut être de 7 lettres directement.

Plus précisément, lorsqu'une coïncidence partielle du texte et du motif est constatée :

$$M_i \neq T_j, \quad M_{i+1} \cdots M_{m-1} = T_{j+1} \cdots T_{m+j-i-1} \tag{1}$$

on décale le motif pour faire coïncider la lettre T_j avec la dernière occurrence $d(T_j)$ de T_j dans M (où d est le dictionnaire renvoyé par l'appel à la fonction `derniere_occurrence`), puis on recommence les comparaisons sur le nouveau facteur du texte. Toutefois, ce décalage n'est fait que si cela fait réellement avancer le motif, c'est-à-dire lorsque $j + d(T_j) > m + j - i - 1$, en d'autres termes si $d(T_j) > m - i - 1$; sinon, on décale le motif de 1 seulement.

Question 10 Exécuter l'algorithme de Boyer-Moore pour la recherche du motif `aababab` dans le texte `aabbbabababacaabbaba`. Même question pour la recherche du motif `ete` dans le texte `la bete a ete etetee`. Compter le nombre de comparaisons effectuées et comparer ce nombre à celui nécessaire dans la recherche naïve et via l'algorithme de Boyer-Moore-Horspool.

Question 11 Implémenter l'algorithme de Boyer-Moore.

La version complète de l'algorithme de Boyer-Moore fait intervenir, en plus de la fonction de dernière occurrence d, une deuxième fonction qui prend en compte le suffixe où la différence entre le motif et le texte a été constatée. Supposons à nouveau qu'on ait une coïncidence partielle (1). Un décalage qui tient compte de cette information va aligner M sur la dernière occurrence du suffixe $u = M_{i+1} \cdots M_{m-1}$ dans M qui est précédée d'une lettre différente de M_i. On note $d_2(i)$ le décalage correspondant. Un code complet de l'algorithme de Boyer-Moore vous sera donné en correction, que vous pourrez alors comparer aux algorithmes précédents. Vous verrez que l'amélioration est faible voire inexistante sur la plupart des exemples de recherche de motif, vis-à-vis de l'algorithme de Horspool, ou même de la version précédente, n'utilisant pas d_2.