Recherche textuelle

Algorithmique

Exercice 1 *QCM d'activation*

Pour chaque question, une seule réponse est correcte.

1. L'algorithme naïf de recherche textuelle compare le motif de longueur m dans un texte de longueur n. Sa complexité dans le pire cas est :

   A. $O(n)$

   B. $O(n \times m)$

   C. $O(n + m)$

   D. $O(m \log n)$

2. L'algorithme de Boyer-Moore améliore la recherche en :

   A. comparant le motif de gauche à droite comme l'algorithme naïf

   B. comparant le motif de droite à gauche et en sautant des positions grâce à des tables de décalage

   C. triant le texte avant la recherche

   D. utilisant la programmation dynamique

3. Dans la table de décalages de Boyer-Moore-Horspool pour le motif `"ABCD"`, quelle est la valeur associée à la lettre `B` ?

   A. 0

   B. 1

   C. 2

   D. 3

4. On recherche le motif `"abc"` dans le texte `"ababcabc"`. L'algorithme naïf trouve le motif aux positions :

   A. 2 uniquement

   B. 2 et 5

   C. 0 et 5

   D. 0, 2 et 5

5. Boyer-Moore est particulièrement efficace lorsque :

   A. le motif est très court (un ou deux caractères)

   B. le motif est long et l'alphabet est grand

   C. le texte et le motif sont identiques

   D. le texte ne contient qu'un seul caractère répété

Exercice 2 *Déroulé à la main de l'algorithme naïf*

On considère `texte = 'abracadabra'` et `motif = 'abr'`.

1. Recopier et compléter le tableau en indiquant, pour chaque position `i`, les comparaisons effectuées et le résultat (échec ou trouvé).
2. Combien de comparaisons au total l'algorithme naïf effectue-t-il ?
3. À quelles positions le motif est-il trouvé ?

Exercice 3 *Table de décalages*

1. Calculer la table de décalages pour le motif `'algorithme'` (longueur 10).
2. Quel est le décalage si le caractère du texte est `'x'` (absent du motif) ?
3. Quel est le décalage si le caractère du texte est `'a'` ?
4. Quel est le décalage si le caractère du texte est `'e'` (dernier caractère du motif, absent ailleurs) ?

Exercice 4 *Déroulé de Boyer-Moore-Horspool*

On cherche le motif `'man'` dans le texte `'le management est un art'`.

1. Calculer la table de décalages du motif.
2. Dérouler l'algorithme étape par étape en indiquant la position `i`, le caractère comparé et le décalage appliqué.
3. Combien d'étapes sont nécessaires ? Combien l'algorithme naïf aurait-il testé de positions ?

Exercice 5 *Compteur de comparaisons*

Modifier la fonction `algo_naif` pour qu'elle renvoie, en plus de la liste des indices, le nombre total de comparaisons effectuées.

Faire de même pour `recherche_boyer_moore`.

Comparer les deux compteurs pour la recherche de `'abd'` dans `'abracadabra'`.

Exercice 6 *Comparaison expérimentale avec `timeit`*

1. Créer une variable `texte` contenant un texte long (par exemple `'a' * 100000`).
2. Choisir un motif de quelques caractères.
3. Utiliser le module `timeit` pour mesurer le temps d'exécution des deux algorithmes.
4. Tester avec un motif plus long (10 caractères). Le rapport de vitesse change-t-il ?

Exercice 7 *Remplacement du slice par une boucle*

Réécrire la comparaison suivante (qui utilise un slice Python) :

```
if texte[i:i + m] == motif:
    indices.append(i)
```

avec une boucle `while` qui compare les caractères **de droite à gauche**, un par un, et qui s'arrête dès qu'un caractère ne correspond pas.

Exercice 8 *Cas limites*

Écrire des assertions pour tester les cas suivants, puis vérifier qu'elles passent pour les deux algorithmes :

1. Le motif est égal au texte entier.
2. Le motif est plus long que le texte.
3. Le texte est vide.
4. Les occurrences se chevauchent (`'aa'` dans `'aaaa'`).
5. Le motif n'apparaît pas du tout.

Exercice 9 *Pire cas de l'algorithme naïf*

1. Construire un texte de longueur 1000 et un motif de longueur 10 qui provoquent le **pire cas** de l'algorithme naïf (un maximum de comparaisons).
2. Vérifier avec le compteur de comparaisons (exercice 4) que le nombre de comparaisons est bien proche de $n \times m$.
3. Tester le même cas avec Boyer-Moore-Horspool. L'algorithme est-il plus rapide ici ?