Informatique *TP*
Algorithmique des textes Ipesup

Introduction

On considère dans tout le sujet un alphabet fini Σ. Un texte est une chaîne $T = T[0]T[1]\cdots T[n-1]$ de longueur n, et un motif est une chaîne $P = P[0]P[1]\cdots P[m-1]$ de longueur m, avec $1 \leq m \leq n$. Une occurrence de P dans T est un indice s tel que

$$T[s+i] = P[i] \quad \text{pour tout } i \in \{0, \ldots, m-1\}.$$

Les indices commencent à 0.

Notation

— Les chaînes sont indexées à partir de 0.
— La longueur d'une chaîne X est notée $|X|$.
— Un facteur de longueur m de T commençant en s est

$$T[s]\cdots T[s+m-1].$$

— Une occurrence de P dans T est un indice s tel que

$$\forall i \in \{0, \ldots, m-1\}, \quad T[s+i] = P[i].$$

— Dans Rabin–Karp, un haché égal impose une vérification explicite des caractères.
— Dans Boyer–Moore, les comparaisons se font de droite à gauche.

Exercice 1

On considère l'algorithme suivant.

Algorithm 1 Recherche naïve

```
1:  procedure RechercheNaive(T, P)
2:      n ← |T|, m ← |P|
3:      L ← [ ]
4:      for s = 0 to n − m do
5:          i ← 0
6:          while i < m and T[s + i] = P[i] do
7:              i ← i + 1
8:          end while
9:          if i = m then
10:             ajouter s à L
11:         end if
12:     end for
13:     return L
14: end procedure
```

On prend

$$T = \texttt{ABABABACABABABABAC}, \qquad P = \texttt{ABABAC}.$$

1. Donner toutes les positions où le motif P apparaît dans le texte T.
2. Exécuter manuellement l'algorithme naïf sur cet exemple. Pour chaque décalage s, indiquer le nombre de comparaisons effectuées.
3. Montrer que l'algorithme renvoie exactement l'ensemble des occurrences de P dans T.
4. Donner une borne supérieure du nombre de comparaisons de caractères dans le pire cas.
5. Construire, pour tout m, un exemple de texte T et de motif P pour lequel cette borne est atteinte à une constante près.
6. Expliquer pourquoi cette méthode devient insuffisante lorsque n est très grand et que l'on effectue de nombreuses recherches de motifs.

Exercice 2

On code les lettres par

$$A = 1, \quad B = 2, \quad C = 3, \quad D = 4.$$

Pour une chaîne $X = X[0] \cdots X[m-1]$, on définit son haché par

$$h(X) = \sum_{i=0}^{m-1} \mathrm{code}(X[i])\, b^{m-1-i} \pmod{q}.$$

Dans cet exercice, on prend $b = 5$ et $q = 23$.

On considère

$$T = \texttt{ABACABADABAC}, \qquad P = \texttt{ABAC}.$$

1. Calculer $h(P)$.
2. Calculer le haché de chacun des facteurs de longueur 4 de T.
3. Retrouver les occurrences de P dans T à l'aide de ces valeurs.
4. On pose
$$H_s = h(T[s] \cdots T[s+m-1]).$$
Montrer que l'on peut calculer H_{s+1} à partir de H_s en temps constant, une fois la valeur $b^{m-1} \bmod q$ connue.
5. Écrire précisément la formule de mise à jour de H_s vers H_{s+1}.
6. Rédiger un pseudo-code complet de Rabin–Karp renvoyant toutes les occurrences exactes du motif.

Exercice 3

Dans cet exercice, on utilise l'alphabet numérique $\Sigma = \{0, 1, \ldots, 9\}$. On considère des motifs de longueur $m = 3$, une base $b = 10$ et un modulo q.

1. Pour $q = 13$, calculer les hachés des chaînes `314`, `159`, `271`, `828`.
2. Trouver, si possible, deux chaînes distinctes de longueur 3 ayant le même haché modulo 13.
3. Expliquer pourquoi l'égalité des hachés ne suffit pas à conclure à l'égalité des chaînes.
4. Modifier le pseudo-code de Rabin–Karp afin de vérifier explicitement les caractères lorsqu'un haché est égal à celui du motif.
5. Montrer que cette version modifiée est correcte, même en présence de collisions.
6. On suppose que les hachés sont uniformément répartis modulo q. Donner une estimation du nombre moyen de vérifications caractère par caractère.
7. Discuter l'influence du choix de q sur les performances de l'algorithme.

Exercice 4

On étudie ici uniquement la règle du mauvais caractère de Boyer–Moore. Les comparaisons entre le motif et le texte se font de droite à gauche.

Pour un caractère $c \in \Sigma$, on note
$$\text{last}(c) = \max\{j \in \{0, \ldots, m-1\} \mid P[j] = c\},$$
avec $\text{last}(c) = -1$ si c n'apparaît pas dans P.

On prend
$$P = \texttt{ABCDABD}.$$

1. Construire la table last pour les caractères A, B, C, D et pour un caractère X absent du motif.
2. On considère
$$T = \texttt{ABCXABCDABCDABD}.$$
Exécuter manuellement Boyer–Moore avec la seule règle du mauvais caractère.
3. À chaque décalage, indiquer :
   (a) les caractères comparés ;

   (b) la position du premier désaccord ;
   (c) le décalage appliqué.
4. Justifier la formule de décalage
$$d = \max(1, j - \text{last}(T[s + j])),$$
lorsque le désaccord se produit en position j du motif.
5. Montrer que ce décalage ne peut pas faire manquer une occurrence du motif.
6. Donner un exemple où la règle du mauvais caractère permet un grand saut.

Exercice 5

On considère maintenant la règle du bon suffixe. Lors d'un désaccord en position j, le suffixe
$$P[j + 1] \cdots P[m - 1]$$
a déjà été reconnu dans le texte. L'idée est de décaler le motif afin de réaligner ce suffixe avec une autre occurrence compatible dans P, ou avec un préfixe de P.

On prend
$$P = \texttt{ANANAS}.$$

1. Lister tous les suffixes non vides de P.
2. Pour chacun des suffixes `S`, `AS`, `NAS`, `ANAS`, déterminer s'il possède une autre occurrence dans P.
3. Déterminer les plus longs suffixes de P qui sont aussi préfixes de P.
4. Expliquer comment la règle du bon suffixe exploite ces informations pour choisir un décalage.
5. Exécuter manuellement Boyer–Moore avec les deux règles sur
$$T = \texttt{BANANANASANANAS}.$$
6. Comparer le nombre de comparaisons effectuées avec celui de la recherche naïve.
7. Donner un exemple de texte et de motif pour lequel Boyer–Moore a un mauvais comportement si les optimisations ne sont pas toutes utilisées.
8. Discuter, sans preuve détaillée, les complexités moyenne et pire cas de Boyer–Moore.

Exercice 6

On souhaite comparer trois méthodes de recherche de motif dans un texte :

recherche naïve, Rabin–Karp, Boyer–Moore.

On considère trois familles de textes :
$$T_1 = \texttt{AAAAAAAAAAAAAAAAAAAA},$$
$$T_2 = \texttt{QWERTYUIOPASDFGHJKLZ},$$
$$T_3 = \texttt{ABRACADABRABRACADABRA}.$$

On considère les motifs
$$P_1 = \texttt{AAAAA}, \qquad P_2 = \texttt{FGH}, \qquad P_3 = \texttt{ABRA}.$$

1. Pour chaque couple (T_i, P_i), prédire qualitativement le comportement de la recherche naïve.
2. Pour chaque couple (T_i, P_i), expliquer si Rabin–Karp est susceptible d'être efficace.
3. Pour chaque couple (T_i, P_i), expliquer si Boyer–Moore est susceptible d'effectuer de grands décalages.
4. Compléter un tableau de comparaison indiquant, pour chaque algorithme :
   (a) son principe ;
   (b) son coût de prétraitement ;
   (c) sa complexité moyenne attendue ;
   (d) sa complexité dans le pire cas ;
   (e) un type d'entrée favorable ;
   (f) un type d'entrée défavorable.
5. Écrire un pseudo-code d'expérience permettant de compter le nombre de comparaisons de caractères effectuées par les trois méthodes.
6. On suppose maintenant que l'on recherche simultanément un grand nombre de motifs de même longueur. Expliquer pourquoi Rabin–Karp peut devenir intéressant.
7. On suppose au contraire que le motif est long et que l'alphabet est grand. Expliquer pourquoi Boyer–Moore est souvent très performant.
8. Conclure en donnant, pour chacun des trois algorithmes, un contexte d'utilisation pertinent.