M2 ATAL | Université de Nantes
X3IT030 - Algorithmique et alignement de chaînes | 2020/2021

Feuille de travaux dirigés
Recherche de Motifs: Algorithmes à base d'index, Transformée de Burrows-Wheeler, FM-index

Exercice 1 (Automate)
Appliquer l'algorithme "recherche à base d'automates" pour trouver toutes les occurrences du motif $P = \texttt{AABAB}$ dans le texte $T = \texttt{AAABABAABAABAB}$.

Exercice 2 (Algorithme de Rabin-Karp – Pré-traitement de P)
On propose une nouvelle méthode pour rechercher de façon exacte un motif P de longueur m dans un texte T de longueur n. Pour simplifier, on supposera dans un premier temps que l'alphabet Σ est le suivant : $\Sigma = \{0, 1, 2 \ldots 9\}$.
Dans ce cas, toute chaîne de k caractères exprimée sur Σ peut se voir comme un *nombre entier* à k chiffres. Par exemple, la chaîne $C =$`658042` peut être vue comme le nombre entier $n_C = 658\,042$.

Étant donné un motif P, on note n_P l'entier correspondant. De même, pour tout $0 \leq i \leq n-m$, on note n_i l'entier correspondant à $T[i \ldots i+m-1]$ (ici, T est donc indicé à partir de 0). Dans ce cas, P apparaît dans T à la position i si et seulement si $n_i = n_P$.

On suppose qu'on dispose d'une fonction `char2int` qui prend en entrée un caractère `c` et fournit en sortie le chiffre qui lui est associé, et que celle-ci s'exécute en temps constant.

1. Écrire en pseudo-code un algorithme `valeur_np` qui permet de calculer n_P *en temps* $O(m)$, et justifier sa complexité.
2. Indiquer la formule qui permet de calculer n_{i+1} à partir de n_i, $T[i]$ et $T[i+m]$.
3. En supposant que la valeur 10^{m-1} a été pré-calculée et stockée, quelle est la complexité en temps du calcul de la Question 2 ?
4. Quelle est la complexité en temps du calcul de la valeur 10^{m-1} ? Pourquoi ?
5. En déduire la complexité en temps nécessaire à l'exécution de l'ensemble des calculs de $n_0, n_1, n_2 \ldots n_{n-m}$.

En s'appuyant sur les réponses précédentes, on peut en déduire un algorithme, qu'on appellera `RK`, dont la complexité en temps est en $O(n+m)$, et qui détermine toutes les positions auxquelles un motif P apparaît dans un texte T.

6. Écrire en pseudo-code l'algorithme `RK`, et justifier sa complexité.

La méthode ci-dessus fonctionne bien car on a supposé que la taille σ de l'alphabet Σ est égale à 10.

7. Comment faire pour adapter cette méthode à n'importe quelle valeur de σ ? Détailler votre réponse.
8. Analyser la complexité en temps de cette nouvelle méthode : (a) si on considère σ non constant, puis (b) si on considère σ constant.

Exercice 3 (Recherche de motif dans plusieurs textes)

On souhaite rechercher un même motif P dans *plusieurs* textes $T_1, \ldots T_k$, en utilisant la méthode de l'arbre des suffixes. On supposera que $|P| = m$, que chaque texte T_i est de longueur n_i, et que $N = \sum_{i=1}^{k} n_i$.

On propose trois façons de procéder :

(a) Concaténer tous les textes $T_1, \ldots T_k$ en un seul texte $\mathcal{T} = T_1 \cdot T_2 \cdot \ldots T_k$, construire $AS(\mathcal{T})$ et rechercher P dans $AS(\mathcal{T})$.

(b) Construire un arbre $AS(T_1, T_2 \ldots T_k)$ qui contient tous les suffixes de T_i, $1 \leq i \leq k$, et rechercher P dans $AS(T_1, T_2 \ldots T_k)$.

(c) Construire k arbres des suffixes différents $AS(T_1)$, $AS(T_2) \ldots AS(T_k)$ et rechercher P dans chacun de ces arbres.

1. Pour chacune de ces méthodes, indiquer (1) sa complexité en temps et (2) ses possibles inconvénients.
2. Illustrer la méthode (b) sur le (petit) exemple suivant : $k = 2$ avec $T_1 = AGACA$, $T_2 = GCAGAG$.

Exercice 4 (Tableau des Suffixes)

1. Construire le tableau des suffixes $TS[]$ du texte $T =$`SENSELESSNESS`.
2. Construire le tableau $LCP[]$ du même texte $T =$`SENSELESSNESS`.
3. Déterminer à partir de $TS[]$ et $LCP[]$ les positions des occurrences du motif `ESS` dans T.
4. Partant de $TS[]$ et $LCP[]$, indiquer une méthode qui permette de construire $AS(T)$, en insérant itérativement les suffixes dans l'ordre où ils apparaissent dans $TS[]$.
5. Illustrer votre méthode sur le texte $T =$`SENSELESSNESS` dont les tableaux $TS[]$ et $LCP[]$ ont été construits aux Questions 1. et 2.
6. Donner les arguments qui montrent que la construction de $AS(T)$ partant de $TS[]$ et $LCP[]$ prend un temps $O(n)$, où $n = |T|$.

Exercice 5 (Recherche de motif approché)

On souhaite trouver une méthode qui réponde au problème de la recherche d'un motif P dans un texte T *à e erreurs près*, et qui repose sur l'arbre des suffixes.
Ici, une erreur est un *mismatch* : autrement dit, on n'autorise que les substitutions de lettres (pas d'insertion ni de suppression). On supposera qu'on travaille sur un alphabet Σ de taille σ.

Par exemple ; pour $P =$ `ATAL`, $T =$ `ATABLETATAETALI` et $e = 2$, on a quatre occurrences de P à e erreurs près dans T, aux positions 1, 6, 8 et 11.

La méthode proposée est la suivante : générer tous les motifs ayant au plus e erreurs par rapport à P, et les chercher, les uns après les autres, dans $AS(T)$ (l'arbre des suffixes de T).

1. Pour un motif P de longueur m, combien y a-t-il de motifs ayant au plus e erreurs par rapport P ? (appelons ce nombre N).
2. Indiquer la valeur de N sur l'exemple donné ci-dessus, dans le cas où $\sigma = 26$ (les 26 lettres majuscules de l'alphabet français).
3. Indiquer, dans le cas général, la complexité en temps de la méthode proposée.

Exercice 6 (Transformée de Burrows-Wheeler 1)

1. Donner la transformée de Burrows-Wheeler (BWT) du texte suivant : `T=trottinette$`, que l'on appellera L (pour "Last", c'est-à-dire la dernière colonne de la matrice de Burrows-Wheeler).
2. Donner la colonne F (pour "First") de la matrice de Burrows-Wheeler sur le même texte.

Exercice 7 (Transformée de Burrows-Wheeler 2)

Supposons que `T=aabaabaabba$`. Rechercher les motifs suivants dans T en utlisant la BWT :

1. `P=ab`
2. `P=bba`
3. `P=aaa`

4. `P=aba`

Exercice 8 (Transformée de Burrows-Wheeler 3)

1. Supposons que l'on connaisse $TS[]$, la table des suffixes d'un texte T (de longueur n). Indiquer une formule générale qui relie $F[i]$ et $TS[i]$ pour tout $1 \leq i \leq n$.
2. Indiquer (en français) un algorithme qui, partant d'un texte T de longueur n, détermine L en temps $O(n)$.

Exercice 9 (Transformée de Burrows-Wheeler 4)

1. Déterminer le texte T qui a donné la BWT suivante : `L=urattg$uuooo`.

On définit les trois tableaux suivants :
— pour tout $0 \leq i \leq n-1$, $rang[i]$ est le nombre d'occurrences du caractère $L[i]$ parmi les positions $0 \leq p \leq i-1$;
— pour tout caractère c de Σ, $first[c]$ est la position dans F de la première occurrence de c ;
— pour tout $0 \leq i \leq n-1$, $L2F[i]$ (pour "Last to First") est la position de la lettre $L[i]$ dans F, en tenant compte nombre d'occurrences de cette lettre.

2. Donner les contenus des tableaux $rang[]$, $first[]$ et $L2F[]$ lorsque `L=urattg$uuooo`.
3. On affirme que pour tout $0 \leq i \leq n-1$, on a $L2F[i] = first[L[i]] + rang[i]$. Justifier.
4. Proposer un algorithme dont la complexité en temps est linéaire, et qui calcule $L2F[]$.
5. Écrire un algorithme qui retrouve T à partir de L, F et $L2F[]$, et donner sa complexité en temps.