

Arbre des suffixes

Eric Rivals

LIRMM

www.lirmm.fr/~rivals



Thème : Structure d'index de texte pour la recherche de motif.

- 1 Structure d'index de texte : notion, généralités
- 2 Arbre des suffixes : présentation, définition et propriétés
- 3 Construction de l'arbre des suffixes :
algorithme d'Ukkonnen
- 4 Applications

Structure d'indexation de texte

Recherche de motif : une autre approche

Soit T un texte de longueur n sur l'alphabet Σ .

Recherche de motif 2 phases :

- 1 pré-traitement du texte T en $O(n)$,
construction d'une structure de données : index de texte
- 2 recherche du motif en $O(m)$ dans l'index

Structure d'index de texte

Caractéristiques d'un **bon** index :

- ① occupation mémoire de l'ordre de $O(n)$
- ② constructible en $O(n)$ unités de temps
- ③ permet la recherche de motif en temps $O(m)$

Trois structures principales :

- ① l'*arbre des suffixes* [Wiener 73, McCreight 76, Ukkonen 92]
- ② la *table des suffixes* [Mamber-Myers 90]
 $O(n \log(n))$, [Kärkkäinen & Sanders 03] $O(n)$
- ③ le *DAWG* (Directed Acyclic Word Graph) [Blumer et al. 85]

Problème

Définition : un *facteur* de T est un mot qui apparaît dans T

$$f \text{ facteur de } T : \exists 1 \leq i \leq j \leq n : T[i, j] = f$$

Problème : Stocker tous les facteurs du texte

Propriété : tout facteur est le préfixe d'un suffixe,
 f préfixe de $\lg(j - i + 1)$ du suffixe $T[i, n]$

Problème : stocker tous les suffixes d'un texte

Solution 1 : arbre des motifs des suffixes de T (voir algorithme d'Aho-Corasick)

L'arbre des suffixes

Exemple : $T := abcabcdaae$, $n := 10$

Position	Suffixe
1	a b c a b c d a a e
2	b c a b c d a a e
3	c a b c d a a e
4	a b c d a a e
5	b c d a a e
6	c d a a e
7	d a a e
8	a a e
9	a e
10	e

Exemple d'arbre des motifs des suffixes de T , appelé *Trie*.

Exemple d'*Arbre des Suffixes* (AS) de T : version compressée du TRIE, où chaque *branche* constituée de nœuds de degré 1 est compressée en un seul arc.

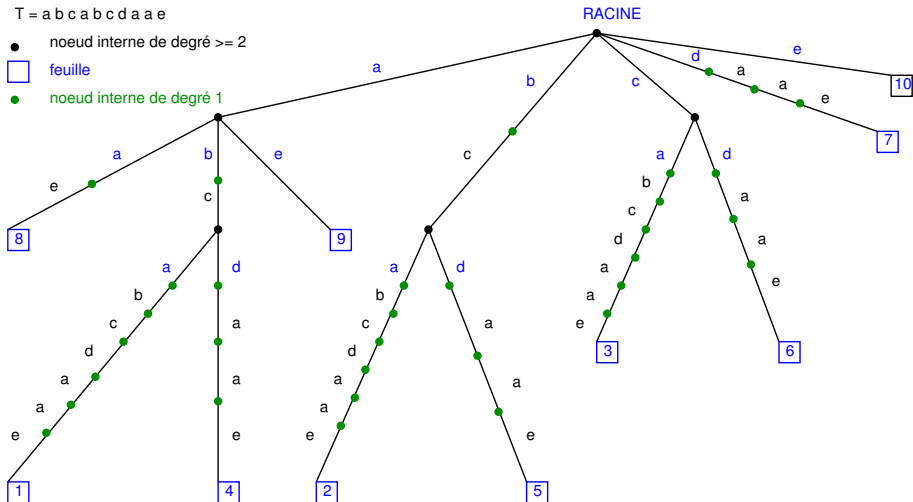
Trie de l'exemple

T = a b c a b c d a a e

● noeud interne de degré ≥ 2

□ feuille

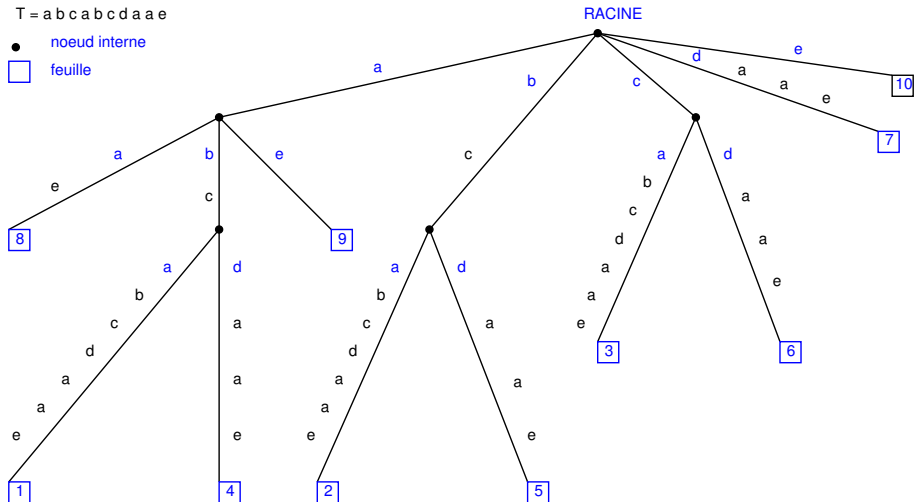
● noeud interne de degré 1



Arbre des suffixes de l'exemple

T = abcabcbdaae

- noeud interne
- feuille



Arbre des suffixes

Définition : un arbre des suffixes (AS) du texte T est un arbre enraciné à n feuilles tel que :

- chaque feuille représente un seul et unique suffixe, celle représentant $T[i, n]$ est étiquetée par i .
- ses arcs sont étiquetés par des mots non vides
- ses nœuds internes sont de degré > 1
- les étiquettes des fils d'un nœud débutent toutes par une lettre différente
- la concaténation des étiquettes sur le chemin de la racine r à une feuille i forme le suffixe $T[i, n]$

Hypothèse : aucun suffixe n'est préfixe d'un autre
condition réalisable en concaténant un nouveau symbole (ex : \$) à T

AS : Propriétés 1

Définition : soit v un nœud . On appelle *facteur de v* le mot épelé sur le chemin de la racine à v . On le note f_v .

Propriétés

- 1 Chaque nœud représente (uniquement) un facteur ; l'inverse est faux.
- 2 À chaque facteur correspond un chemin unique débutant à la racine de l'AS
- 3 qui s'arrête à un point unique sur un arc.
- 4 Le nombre et les étiquettes des feuilles du sous-arbre de v donnent le nombre et les positions des occurrences de f_v .
- 5 Le nœud qui est plus proche ancêtre commun à deux nœuds représente leur plus long préfixe commun.

AS : Propriétés 2

Propriété : un facteur f est associé à chemin unique.

Preuve : On descend dans l'arbre à partir de r en épelant f . À chaque nœud on choisit l'arc ayant comme premier caractère le caractère courant de f . Comme les arcs sont étiquetés par des premiers caractères différents, l'arc emprunté est unique et par induction le chemin est unique.

Propriété : relation de v à f_v , **un nœud représente un unique facteur**
Il représente le préfixe commun de tous les suffixes qui ont leur feuille dans le sous-arbre de v .

Remarques :

- 1 accès facile au premier caractère d'un arc
- 2 test " f est un facteur de T " est différent de l'obtention des positions de ses occurrences.

Recherches rapide et lente

Deux cas différents :

Rapide : recherche des occurrences d'un facteur f du texte.

Descente rapide dans l'AS pour rechercher l'unique point représentant le facteur ; on sait que ce point existe puisque f est un facteur de T . Complexité proportionnelle au nombre de nœuds sur le chemin.

Lente : recherche d'un motif f dont on ne sait pas s'il apparaît dans le texte. La descente dans un arc oblige à vérifier que le label de l'arc correspond aux caractères courants dans f . Complexité proportionnelle à la taille du motif.

Construction d'un arbre des suffixes

Construction simple de l'AS

Deux possibilités :

Option 1 construction incrémentale d'un arbre compressé :

L'arbre initial contient la racine et l'arbre final est l'AS complet.

Option 2 Construire l'arbre des motifs des suffixes et le compresser.

Complexité temporelle en $O(n^2)$.

Construction incrémentale de l'AS

- **Idée** : construction incrémentale d'un arbre compressé dans lequel on insère les suffixes de $T[1, n]$ à $T[n, n]$. L'arbre initial contient la racine et l'arbre final est l'AS complet. À chaque étape on crée un arbre intermédiaire en insérant le suffixe courant à l'arbre précédent.
- **Étape i** : Pour un suffixe $T[i, n]$, descendre dans l'arbre courant en recherchant son plus long préfixe déjà stocké. Puis au point d'arrêt de la descente, si f est le facteur courant alors il existe un facteur g tq : $T[i, n] = fg$. La descente se termine
 - 1 soit sur un nœud v : alors $f_v = f$, créer une feuille fille de v et étiqueter son arc par g ;
 - 2 soit au milieu d'un arc sortant d'un nœud w : décomposer l'arc en 2 en créant un nœud v fils de w et tq $f_v = f$, puis (idem que 1) créer une feuille fille de v et étiqueter son arc par g .

Algorithme de construction d'Ukkonen

Définition : *Arbre des Suffixe Implicite*

Un *Arbre des Suffixe Implicite* (ASI) est un AS sans la contrainte que chaque suffixe soit associé à une feuille.

Idée

Construction incrémentale de l'AS implicite de T , puis transformation en AS.

Lien suffixe

Soit un nœud interne v d'un ASI tq $f_v = a\alpha$, $a \in \Sigma$, $\alpha \in \Sigma^*$. On appelle *lien suffixe*, le lien de v vers le nœud représentant le facteur α s'il existe.

Complexité en temps et espace

Construction en temps $O(n)$ et dite « on-line » par [Ukkonen 92].

Algorithme d'Ukkonen 1

Idée

Construction d'une suite I_1, \dots, I_n d'ASI pour les préfixes de $+$ en $+$ longs : $T[1, 1], \dots, T[1, n]$.

Cela constitue les n *phases* de l'algorithme

Phase

La construction de I_i à partir de I_{i-1} , est nommée *phase* i , pour i allant de 1 à n .

La phase i insère le préfixe $T[1, i]$ dans l'ASI I_{i-1} pour créer I_i .

extensions

La phase i comprend i *extensions*, numéroté par j pour $j \in 1, \dots, i$.
L'extension j insère le suffixe $T[j, i]$ du préfixe courant dans l'ASI.

Ukkonen : exemple

Exemple : $T := abcabcdaae$, $|T| := 10$

ASI initial : l_0 contient uniquement la racine

Phase 1 : insère $T[1, 1] := a$ dans l_0 en 1 extension, résultat : l_1
E(1,1) : insère réellement $T[1, 1]$ dans l_0

Phase 2 : insère $T[1, 2] := ab$ dans l_1 en 2 extensions, résultat : l_2
E(2,1) : insère $T[1, 2]$ dans l_1
E(2,2) : insère $T[2, 2]$ dans l_1

Phase 3 : insère $T[1, 3] := abc$ dans l_2 en 3 extensions, résultat : l_3
E(3,1) : insère $T[1, 3] := abc$
E(3,2) : insère $T[2, 3] := bc$
E(3,3) : insère $T[3, 3] := c$
⋮

Exemple suite

Phase 9 : insère $T[1, 9] := abcabcdaa$ dans I_8 en 9 ext.

E(9,1) : insère $T[1, 9] := abcabcdaa$

E(9,2) : insère $T[2, 9] := bcabcdaa$

E(9,8) : insère $T[8, 9] := aa$

E(9,9) : insère $T[9, 9] := a$

Phase 10 : insère $T[1, 10] := abcabcdaae$ dans I_9 en 10 ext.

E(10,1) : insère $T[1, 10] := abcabcdaae$

E(10,2) : insère $T[2, 10] := bcabcdaae$

E(10,8) : insère $T[8, 10] := aae$

E(10,9) : insère $T[9, 10] := ae$

E(10,10) : insère $T[10, 10] := e$

Résultat : I_{10} . Puis transformation de I_{10} en AS.

Algorithme d'Ukkonen 2

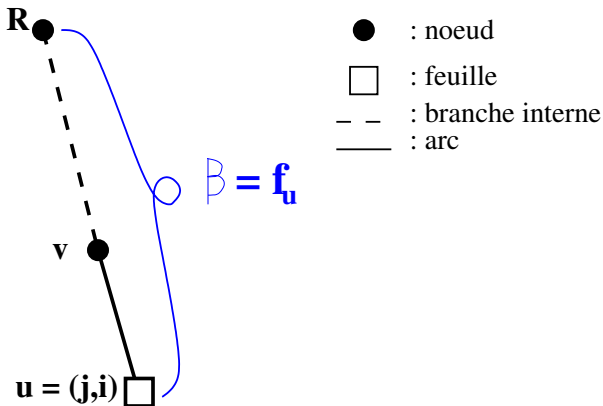
Extension j de la phase $i + 1$:

Insertion de $T[j, i + 1]$ sachant que $T[j, i]$ est déjà dans I_i .
Similaire à l'insertion d'un suffixe dans la méthode simple :
trouver le point qui représente $T[j, i]$ (point d'arrêt de la descente dans I_i).

Dépendance entre extensions $j - 1$ et j : le point d'arrêt pour $T[j, i + 1]$ peut être trouvé à partir du point d'arrêt de $T[j - 1, i + 1]$ en suivant le lien suffixe. Ce lien suffixe existe puisque $T[j, i]$ et $T[j - 1, i]$ sont dans l'arbre.

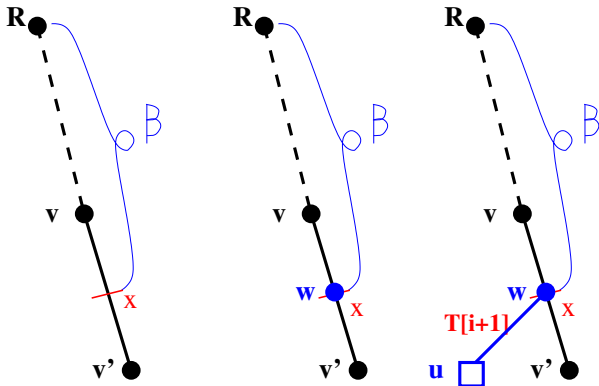
3 cas possibles : soit le point d'arrêt est une feuille (*Règle 1*), soit il est à l'intérieur de l'arbre (indifféremment sur un nœud ou sur un arc). Pour cette dernière éventualité, on distingue deux sous-cas : le caractère suivant sur la branche est différent de $T[i + 1]$ (*Règle 2*) ou bien il est égal à $T[i + 1]$ (*Règle 3*).

Règle 1



Transformation : $f_u \leftarrow f_u.T[i + 1]$
 ou bien : $u \leftarrow (j, i + 1)$
 ou encore : $e \leftarrow e + 1$

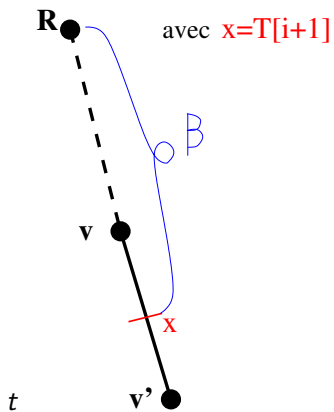
Règle 2



avec $x \neq T[i+1]$

Transformation : créer nœud intermédiaire w
 puis créer feuille u fille de w
 avec un arc étiqueté $T[i+1]$

Règle 3



Transformation : rien à faire

Création des liens suffixes

Propriété Lien Suffixe (LS)

Soient $a \in \Sigma$ et $u \in \Sigma^*$.

- Si un nœud v de label au est créé à l'extension j de la phase $i + 1$, alors au plus tard un nœud de label u sera créé à la fin de l'extension $j + 1$ de cette phase.
- En fin de phase, tout nœud interne dans I_i a un lien suffixe.
- Tout nœud interne d'un ST implicite a un lien suffixe.
- Première extension, le chemin pour $T[1, i]$ se termine à la feuille 1.
- Seconde, débute sur le père de la feuille 1.

Algorithme d'Ukkonen 3

Remarque : « Once a leaf, always a leaf »

Aucune procédure ne modifie les feuilles dans cet algorithme.
Une feuille reste donc une feuille.

Propriété 1

Création d'une feuille par la Règle 2 dans l'extension j de la phase i implique l'utilisation de la Règle 1 à l'extension j de la phase $i + 1$.

Preuve

$E(i,j)$ crée une feuille j pour $T[j, i]$, donc la recherche du point d'arrêt pour $T[j, i + 1]$ aboutit à la feuille j . D'où la Règle 1. CQFD

Extension 1 et 2 d'une phase

Première extension de la phase $i + 1$

Le nœud d'arrêt est la feuille qui stocke $T[j, i]$, c.-à.-d. le plus long suffixe inséré dans I_j . C'est forcément une feuille ; on la mémorise d'une phase sur l'autre.

On utilise la règle 1

Cela prend $O(1)$ en temps

On mémorise le père v de cette feuille pour $E(i+1, 2)$.

Deuxième extension de la phase $i + 1$

Le nœud v existait dans I_j , donc il a un lien suffixe (Prop. LS) qui pointe sur $s(v)$. Que $s(v)$ soit la racine ou un nœud interne, il stocke un préfixe de $T[2, i + 1]$. On recherche le point d'arrêt à partir de $s(v)$ et pas nécessairement de la racine.

Algorithme d'Ukkonen 4

Contenu d'une feuille : Feuille annotée (j, e) où e est une variable indiquant i à la phase i (la fin du suffixe courant).

Extension implicite : À chaque phase, on incrémente e de 1 on effectue ainsi toutes les extensions pour lesquelles la Règle 1 est utilisée.

Propriété 2 : les indices de ces extensions à chaque phase forment un intervalle $[1, k_i]$ où k_i croît avec i .

Propriété 3 : la Règle 3 est une règle « stop » : la phase se termine avec la 1ère extension qui l'utilise.

Complexité en temps

Analyse amortie de la complexité.

Profondeur de nœud

Pour un nœud v , c'est le nb de nœuds entre la racine et v .

Propriété profondeur

La profondeur du nœud courant reste dans l'intervalle $[0, n]$.

Elle remonte au plus de deux unités par extension.

En conséquence, le nombre d'incrément est bornée par $3n$.

Transformation d'un ASI en Arbre des suffixes

se fait par un parcours de l'arbre en $O(n)$.

Théorème

Pour construire l'arbre des suffixes d'un texte T de longueur n , l'algorithme d'Ukkonen requière $O(n)$ temps si l'alphabet est de cardinal constant.

Applications

Applications des AS

Définition

Un *arbre des suffixes généralisé* est un AS adapté pour contenir un ensemble de séquences.

Applications :

Recherche de plusieurs motifs : avec un nombre indéfini de motifs

Recherche du motif complémentaire, complémentaire renversé ou renversé

Plus longue sous-chaîne commune à 2, voire x séquences

Détermination de contamination d'ADN par des vecteurs viraux

Statistique de texte

Compression de texte

Identification des répétitions maximales

Recherche à k substitutions près

Soient M un motif de m caractères, caractères, k un entier tel que $k \leq m$ et $h(U, V)$ la distance de Hamming entre deux chaînes de caractères U, V .

Définition : trouver les indices de début i de tous les facteurs M' de T tels que $h(M, M') \leq k$

Procédure : recherche par descente dans l'AS à partir de R

```

1  $c = 0$ ;
2 tant que (motif non trouvé) et ( $c \leq k$ ) faire
3   descendre l'AS à partir du nœud courant jusqu'à mismatch;
4   si lg du facteur du nœud courant  $\geq m$  alors
5     facteur trouvé aux positions des indices des feuilles;
6     du sous arbre;
7   sinon
8      $c = c + 1$ ;

```