Des arbres de suffixes aux vecteurs de suffixes

Élise Prieur[1] and Thierry Lecroq[1]

ABISS, Universié de Rouen, 76821 Mt-St-Aignan Cedex,
{elise.prieur,thierry.lecroq}@univ-rouen.fr

Résumé *Nous proposons deux algorithmes de passage d'un arbre de suffixes à un vecteur de suffixes et inversement. Un vecteur de suffixes est une structure de données linéaire équivalente à l'arbre des suffixes. Il a été introduit par Monostori [1,2,3]. Nous augmentons cette structure d'un compteur du nombre d'occurrences pour les facteurs reconnus par les nœuds. Ceci nous a permis de créer un outil de comptage des facteurs répétés en fonction de leurs longueurs.*

Keywords: arbres de suffixes, vecteurs de suffixes, répétitions.

1 Introduction

Un vecteur de suffixes d'un mot y constitue une structure de données alternative à un arbre de suffixes. Il permet de stocker les mêmes informations dans un espace moindre. Les vecteurs de suffixes ont été introduits par Monostori [1,2,3] pour détecter des cas de plagiats. Le vecteur de suffixes du mot y consiste en une suite de boîtes placées à certaines positions sur le mot y (voir Figure 1). Ces boîtes contiennent les mêmes informations que les nœuds de l'arbre de suffixes de y. Monostori a donné un algorithme de construction linéaire « on-line » d'un vecteur non compact de suffixes et un algorithme linéaire de compaction d'un vecteur de suffixes. Nous décrivons ici deux algorithmes linéaires de conversion d'un arbre de suffixes à un vecteur non compact et inversement. L'intérêt des vecteurs de suffixes, outre leur taille réduite, réside dans la lecture linéaire de la structure ce qui permet de détecter plus facilement les répétitions que dans les arbres de suffixes. Afin d'exploiter cette propriété, nous montrons comment augmenter les boîtes des vecteurs d'un compteur pour calculer les répétitions.

2 Notations

On considère un alphabet fini A. On note $\mathcal{A}_C(\mathit{Suff}(y))$ l'arbre de suffixes du mot $y \in A^*$ de longueur n auquel on a ajouté un terminateur. Les branches de $\mathcal{A}_C(\mathit{Suff}(y))$ sont étiquetées par des couples (*position*, *longueur*). On identifie chaque nœud p de l'arbre $\mathcal{A}_C(\mathit{Suff}(y))$ avec la concaténation des branches de l'unique chemin depuis la racine jusqu'à p. On note $\delta(p,(i,\ell)) = q$ lorsqu'une branche, étiquetée par (i,ℓ), relie le nœud p au nœud q. On note $\textsc{Cible}(p,a)$, s'il existe, l'état q pour lequel a est la première lettre de l'étiquette de la branche de p à q. Le lien suffixe d'un nœud au est $s(au) = u$, avec $a \in A$ et $u \in A^*$. On note $\mathit{posd}(u,y)$ la position droite de la première occurrence du mot u dans le mot y, quand u est un facteur de y.

3 De l'arbre au vecteur

Soit $u \in \mathcal{A}_C(\mathit{Suff}(y))$ un nœud de l'arbre des suffixes du mot y. Soit $j = \mathit{posd}(u,y)$ la position droite de la première occurrence de u dans y. Alors dans le vecteur de suffixes de y, il y a une boîte B_j en position j. Dans

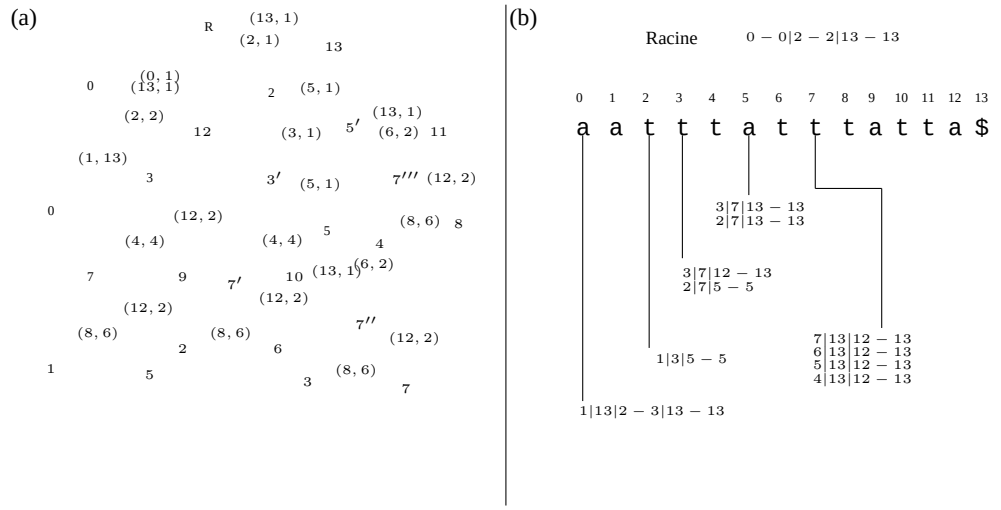

FIG. 1. (a) Arbre de suffixes de `aatttatttatta$`. (b) Vecteur non compact de suffixes de `aatttatttatta$`. Les liens suffixes sont indiqués en pointillé. Un vecteur peut être compacté lorsque les transitions des lignes d'une boîte sont incluses dans les transitions d'une autre ligne d'une même boîte. On crée alors une boîte réduite. C'est, par exemple, le cas ici pour les 4 lignes de la boîte à la position 7 qui peuvent n'en faire qu'une, ainsi que les deux lignes de la boîte à la position 5.

B_j il existe une ligne h telle que $B_j[h,0] = |u|$, $B_j[h,1] = \mathit{posd}(\textsc{Cible}(u, y[j+1]), y)$ et $\forall a \in A \setminus \{y[j+1]\}$ telle que $\textsc{Cible}(u,a)$ est définie on a une transition $L[i] \in B_j[h,2]$ telle que $L[i].f = \mathit{posd}(\textsc{Cible}(u,a), y)$ et $L[i].d = L[i].f - |\textsc{Cible}(u,a)| + 1 + |u|$ (voir FIG. 1). Le traitement de la racine peut se faire de manière analogue.

4 Du vecteur à l'arbre

Soit B_j la boîte du vecteur de suffixes du mot y à la position j. La boîte B_j est en fait un tableau à ℓ lignes et trois colonnes. La première colonne contient les profondeurs, la deuxième les chemins naturels et la troisième les listes de transitions : $B[h,0] = \mathit{prof}$, $B[h,1] = \mathit{cn}$, $B[h,2] = L$ pour $0 \leqslant h \leqslant \ell - 1$. Le chemin naturel d'un nœud p de l'arbre $\mathcal{A}_C(\mathit{Suff}(y))$ associé à un facteur $y[d..f]$ tel que $d = \mathit{posd}(y[d..f], y)$ est la position de la boîte associée au nœud q tel que $\textsc{Cible}(p, y[f+1]) = q$. Chaque transition $L[i]$ est stockée sous la forme d'un couple (d,f) (on utilise les notations $d = L[i].d$ et $f = L[i].f$). Soit $u = y[j - \mathit{prof} + 1..j]$, alors cela implique $u \in \mathcal{A}_C(\mathit{Suff}(y))$, et $y[j - \mathit{prof} + 1..\mathit{cn}] \in \mathcal{A}_C(\mathit{Suff}(y))$ et $\delta(u, (j+1, \mathit{cn} - j)) = y[j - \mathit{prof} + 1..\mathit{cn}]$ et $uy[L[i].d..L[i].f] \in \mathcal{A}_C(\mathit{Suff}(y))$ et $\delta(u, (L[i].d, L[i].f - L[i].d + 1)) = uy[L[i].d..L[i].f]$ pour $0 \leqslant i \leqslant |L| - 1$. De plus on a $B_j[h,0] = B_j[h+1,0] + 1$ pour $0 \leqslant h \leqslant \ell - 2$ [3] ce qui implique, par définition des liens suffixes que $s(y[j - B_j[h,0] + 1..j]) = y[j - B_j[h+1,0] + 1..j]$ pour $0 \leqslant h \leqslant \ell - 2$ (voir FIG. 1). Le traitement de la racine peut se faire de manière analogue.

5 Exemple

L'arbre et le vecteur de suffixes de la séquence $y =$ `aatttatttatta$` sont représentés sur la FIG. 1. On considère la première ligne de la boîte en position 7 du vecteur. On voit que $B_7[0,0] = 7$ donc cette

ligne représente un facteur de longueur 7 soit $y[7-7+1..7] = \texttt{atttatt}$ équivalent au nœud 7 dans $\mathcal{A}_C(\textit{Suff}(\texttt{aatttatttatta\$}))$. Nous avons ensuite $B_7[0,1] = 13$, donc le chemin naturel de ce nœud défini par CIBLE($\texttt{atttatt}, y[8]$) nous amène à un nœud en position 13. Ceci est équivalent à la transition $\delta(\texttt{atttatt}, (8,6))$ de l'arbre, allant du nœud 7 à la feuille 1 représentant le suffixe $y[7-7+1..13]$ soit $\texttt{atttatttatta\$}$. La dernière colonne $B_7[0,2] = ((12,13))$ correspond à la transition $\delta(\texttt{atttatt}, (12,2))$ de l'arbre, allant du nœud 7 à la feuille 5 correspondant au suffixe $\texttt{atttatt}y[12..13] = \texttt{atttatta\$}$.

À présent, voyons comment parcourir un vecteur de suffixes. Soit $x = \texttt{tatt}$, on cherche à savoir si x est un facteur de y. On examine d'abord la racine du vecteur, on y trouve la transition $(2,2)$ comme $y[2] = \texttt{t}$ on la suit. On atteint la boîte B_2, comme $y[3] \neq \texttt{a}$, on ne peut pas suivre le chemin naturel. On regarde donc dans $B_2[0,2]$ la transition $(5,5)$, $y[5] = \texttt{a}$ donc on la suit. Dans la boîte B_5 on considère la ligne j telle que $B_5[j,0] = 2$ car le préfixe déjà lu est $\texttt{ta}$. Comme $y[6] = \texttt{t}$, on poursuit par le chemin naturel c'est-à-dire par le facteur $y[6..7]$ égal à $\texttt{tt}$ donc x est un facteur de y.

6 Comptage

À chaque ligne d'une boîte correspond un nœud u de l'arbre (voir section 4) pour lequel on pose $nbOcc(u)$ son nombre d'occurrences. Soit $nf(t)$ le nombre de feuilles provenant de la transition t tel que $nf(t) = 1$ si $t.f = n$; $nf(t) = nbOcc(v)$ sinon (avec v tel que $\textit{posd}(v,y) = t.f$). On a alors $nbOcc(u) = \sum_L nf(L[i])$. Nous en avons déduit un algorithme linéaire pour compléter chaque ligne du vecteur avec $nbOcc$.

Soit $lg < n$ la longueur des facteurs dont on souhaite connaître le nombre de répétitions, et $lpocc$ la liste des couples (*position gauche*, *nombre d'occurrences*). Pour chaque nœud tel que $B_j[h,0] \geqslant lg$, on met à jour $lpocc$ en utilisant $nbOcc$ de la ligne h de la boîte en position j. Ceci nous permet de donner le nombre d'occurrences de chaque facteur de y de longueur lg.

7 Perspectives

Nos objectifs sont :
- affiner l'implantation des vecteurs compacts de suffixes de Monostori pour les séquences nucléotidiques puisque les résultats pratiques de Monostori montrent que sa représentation est plus efficace pour des grands alphabets que pour des petits ;
- proposer un algorithme direct de construction des vecteurs compacts de manière à traiter de longues séquences ;
- utiliser l'outil de comptage pour la détection de répétitions exactes dans des séquences biologiques afin d'aider à la détection des éléments transposables.

Références

[1] K. Monostori, A. B. Zaslavsky, I. Vajk. Suffix Vector : A Space-Efficient Suffix Tree Representation. In *Proceedings of the 12th International Symposium on Algorithms and Computation*, P. Eades and T. Takaoka eds., volume 2223 of *Lecture Notes in Computer Science*, pages 707-718, Christchurch, New Zealand, 2001. Springer Verlag.

[2] K. Monostori, A. B. Zaslavsky, H. W. Schmidt. Suffix Vector : Space- and Time-Efficient Alternative to Suffix Trees. In *CRPITS '02 : Proceedings of the 25th Australasian Computer Science Conference*, volume 4, pages 157-165, Melbourne, 2002. Australian Computer Society, Inc.

[3] K. Monostori. *Efficient Computational Approach to Identifying Overlapping Documents in Large Digital Collections*. PhD Thesis, Monash University, 2002.