Construction directe d'un vecteur de suffixes compact et répétitions maximales

Élise PRIEUR

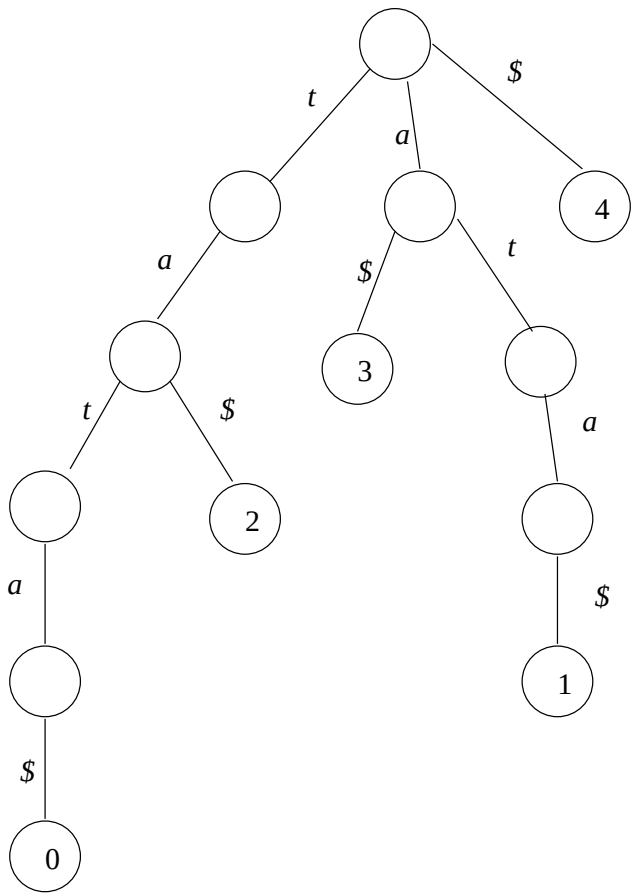
LITIS - Université de Rouen

20 avril 2006

Contexte général

- analyse de séquences (répétitions dans les séquences biologiques, ...) ;
- accès rapide aux facteurs de la séquence ;
- structures pour l'indexation.

« Trie » pour tata$

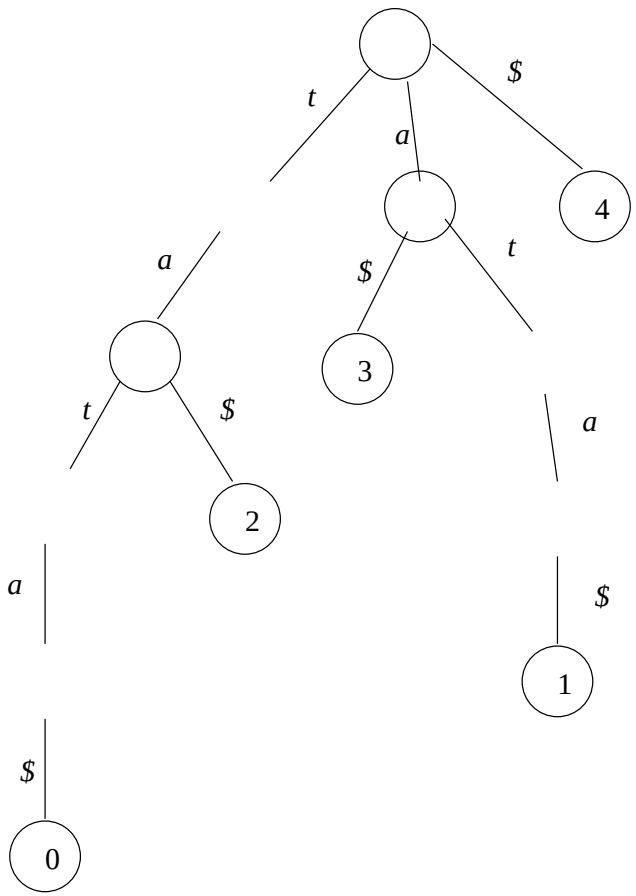


Les suffixes de tata$

0	1	2	3	4	
t	a	t	a	$	0
	a	t	a	$	1
		t	a	$	2
			a	$	3
				$	4

- L'arbre de suffixes d'une séquence est une structure d'index permettant de stocker avec un accès rapide tous les facteurs de cette séquence.
- Les feuilles représentent tous les suffixes de la séquence.

« Trie » pour tata$

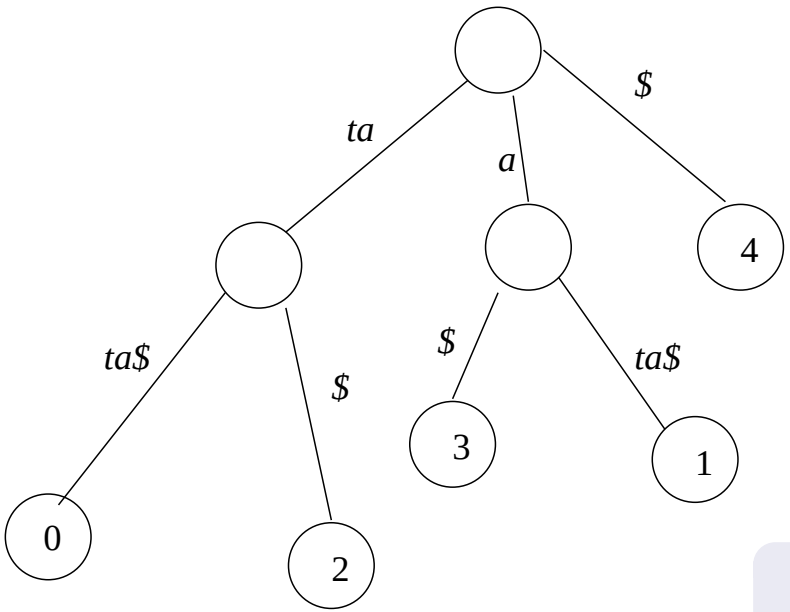


Les suffixes de tata$

0	1	2	3	4	
t	a	t	a	$	0
	a	t	a	$	1
		t	a	$	2
			a	$	3
				$	4

- L'arbre de suffixes d'une séquence est une structure d'index permettant de stocker avec un accès rapide tous les facteurs de cette séquence.
- Les feuilles représentent tous les suffixes de la séquence.

« Trie » pour tata$

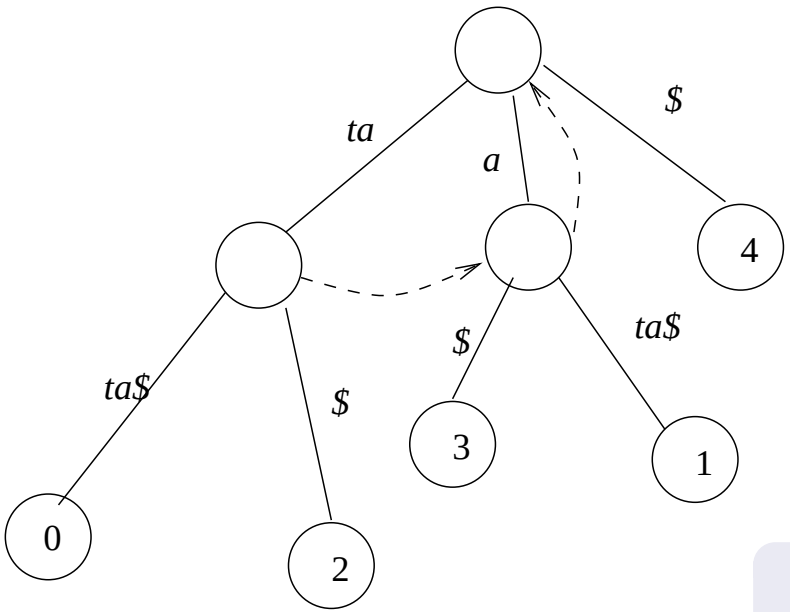


Les suffixes de tata$

0	1	2	3	4	
t	a	t	a	$	0
	a	t	a	$	1
		t	a	$	2
			a	$	3
				$	4

- L'arbre de suffixes d'une séquence est une structure d'index permettant de stocker avec un accès rapide tous les facteurs de cette séquence.
- Les feuilles représentent tous les suffixes de la séquence.

« Trie » pour tata$

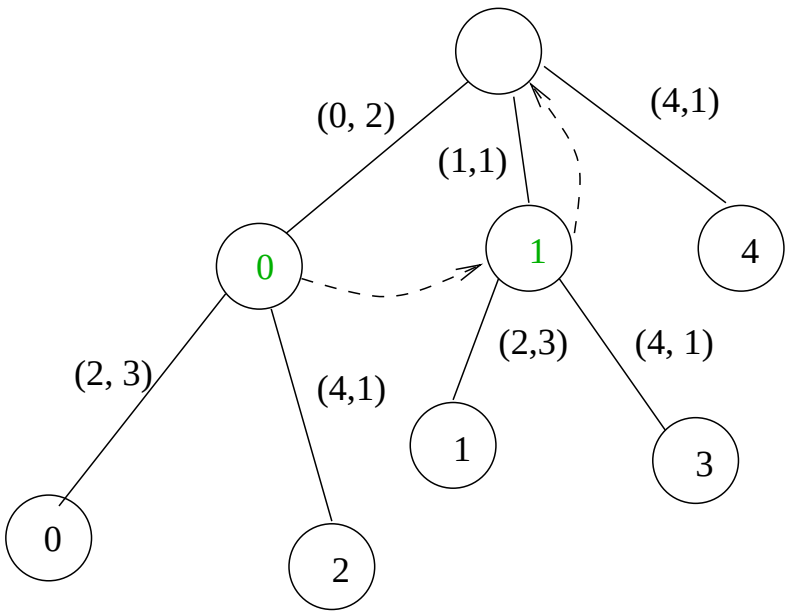


Les suffixes de tata$

0	1	2	3	4	
t	a	t	a	$	0
	a	t	a	$	1
		t	a	$	2
			a	$	3
				$	4

- L'arbre de suffixes d'une séquence est une structure d'index permettant de stocker avec un accès rapide tous les facteurs de cette séquence.
- Les feuilles représentent tous les suffixes de la séquence.

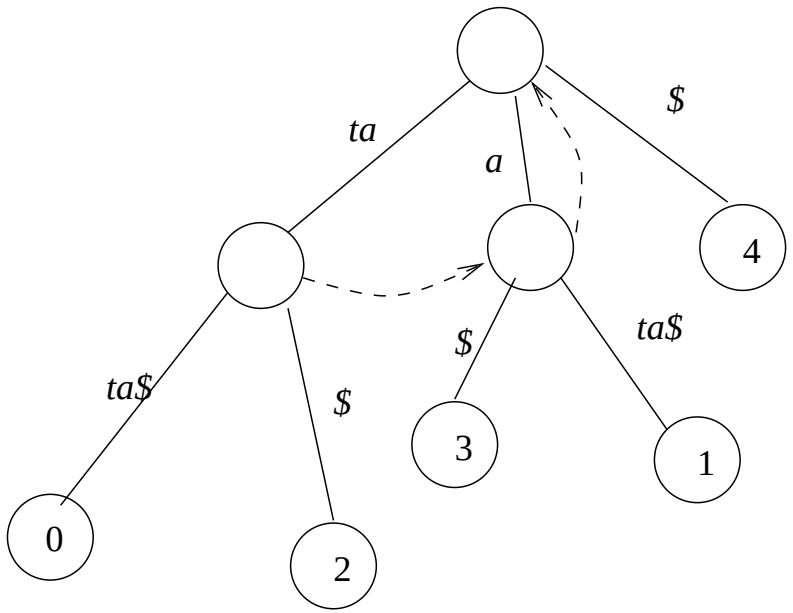
Arbre des suffixes pour tata$



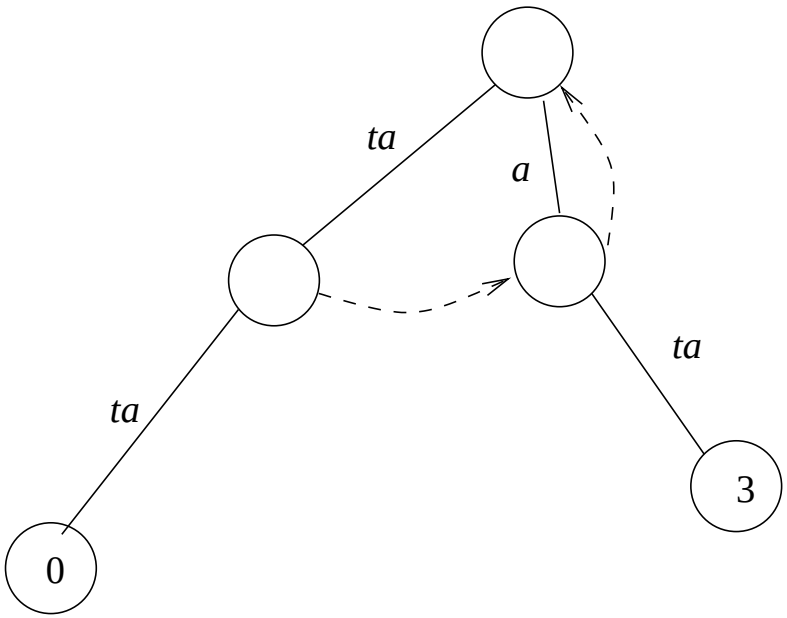
Les suffixes de tata$

0	1	2	3	4	
t	a	t	a	$	0
	a	t	a	$	1
		t	a	$	2
			a	$	3
				$	4

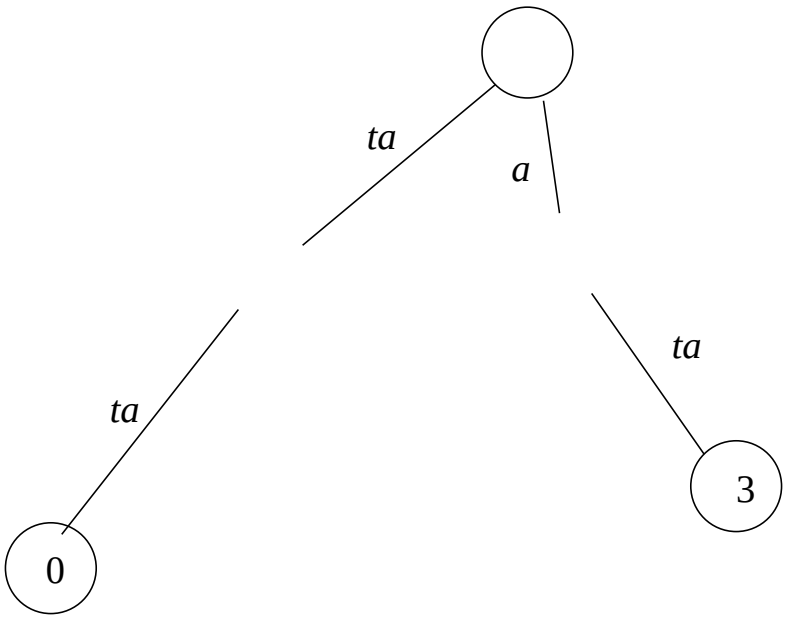
Arbre implicite pour tata



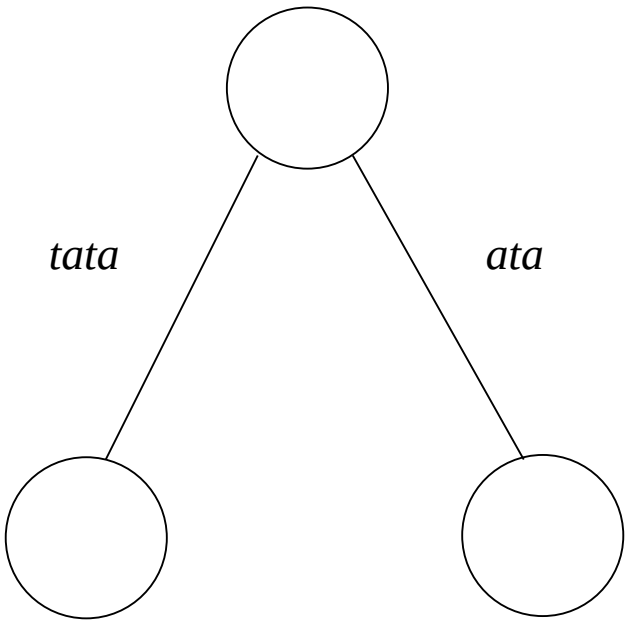
Arbre implicite pour tata



Arbre implicite pour tata



Arbre implicite pour tata



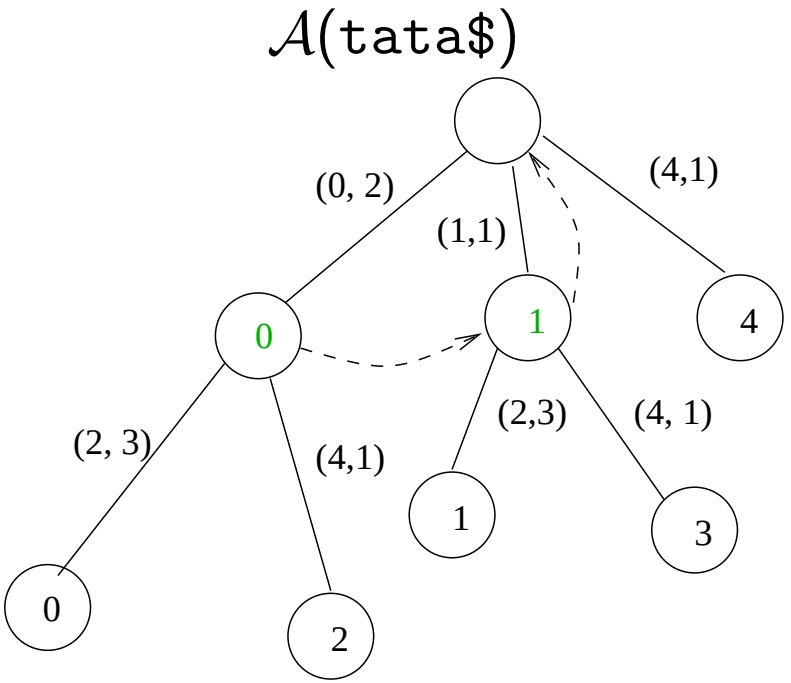
Bref historique

- Patricia Trie (1968) ;
- Weiner (1973) ;
- McCreight (1976) ;
- Ukkonen (1995) ;
- Farach (1997) ;
- tables de suffixes
  - Manber et Myers, $O(nlog(n))$, 1991 ;
  - Kärkkäinen et Sanders, $O(n)$, 2003 ;
  - Ko et Aluru, $O(n)$, 2003 ;
  - Kim, Sim, Park et Park, $O(n)$, 2003.
- vecteurs de suffixes (2001) ;
- ...

Exemple

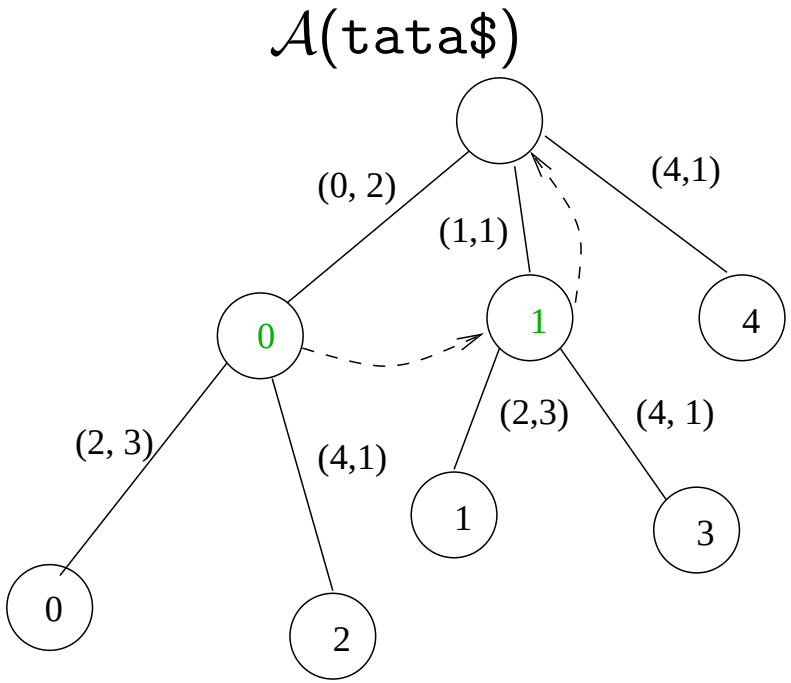
Considérons le facteur $u = \texttt{ta}$ de la séquence $y = \texttt{tata\$}$:

- sa première occurrence débute à la position 0 et termine à la position 1
- le nœud 0 représente u $\delta(\texttt{ta}, (2, 3)) = \texttt{tata\$}$ (feuille 0)
- et $\textsc{Cible}(\texttt{ta}, \texttt{t}) = \texttt{tata\$}$.



Notations

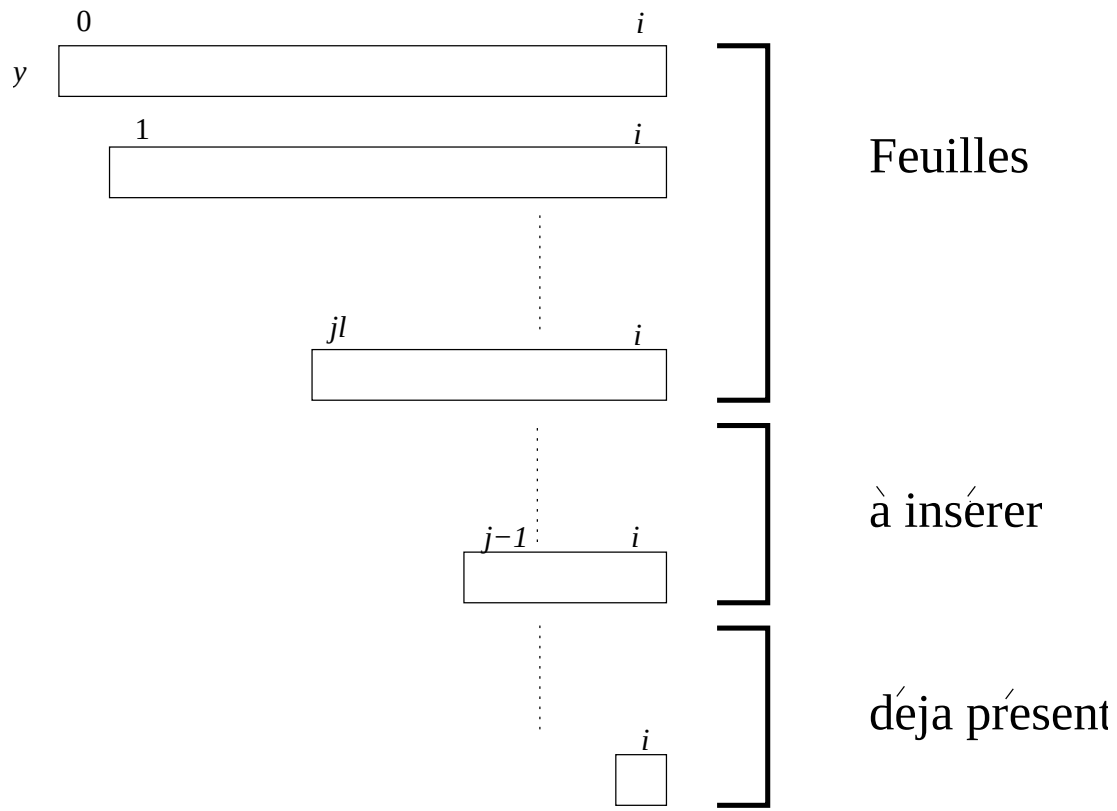
- A un alphabet fini ;
- $\mathcal{A}(y)$, $\mathcal{V}(y)$: l'arbre de suffixes et le vecteur de suffixes d'un texte $y \in A^*$ de longueur n ;
- les nœuds identifiés aux facteurs ;
- $\delta(p, (i, \ell)) = q$: branche du nœud p au nœud q étiquetée (i, ℓ) ;
- $\textsc{Cible}(p, a)$: nœud q tel qu'il existe une branche de p à q dont l'étiquette commence par a ;



L'algorithme d'Ukkonen

- Algorithme « on-line »
- construction séparée en n phases elles-mêmes scindées en extensions
- Lors de la phase i, construction de l'arbre implicite de $y[0..i]$ à partir de celui de $y[0..i-1]$ en ajoutant dans l'arbre tous les suffixes de $y[0..i]$
- Lors de l'extension j de la phase i, le suffixe $y[j+1..i]$ est inséré dans l'arbre.
- Le dernier facteur inséré est noté $w = y[j+1..i-1]$.

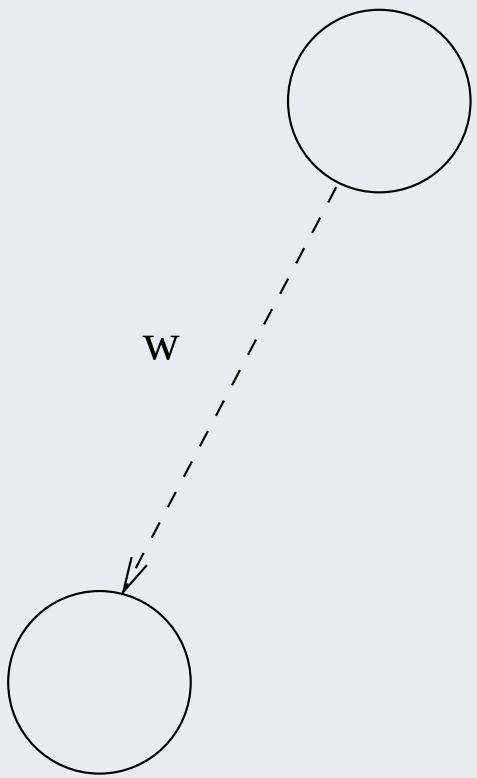
L'algorithme d'Ukkonen



Les 3 règles

L'algorithme d'Ukkonen est basé sur 3 règles formulées par Gusfield :

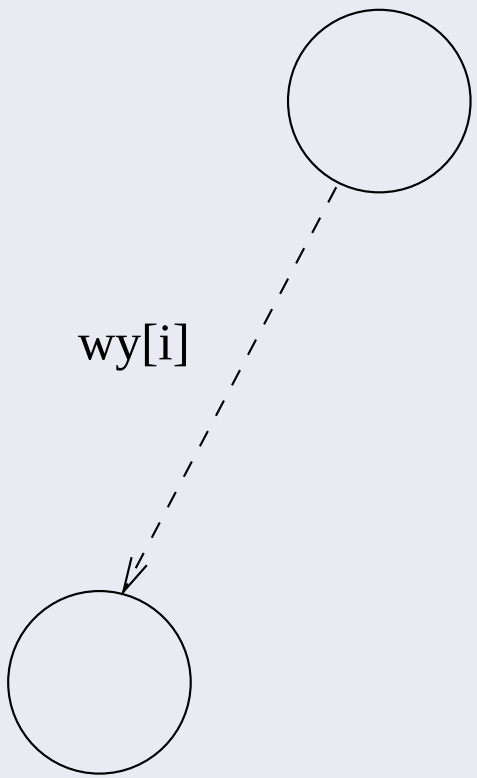
Règle 1



Les 3 règles

L'algorithme d'Ukkonen est basé sur 3 règles formulées par Gusfield :

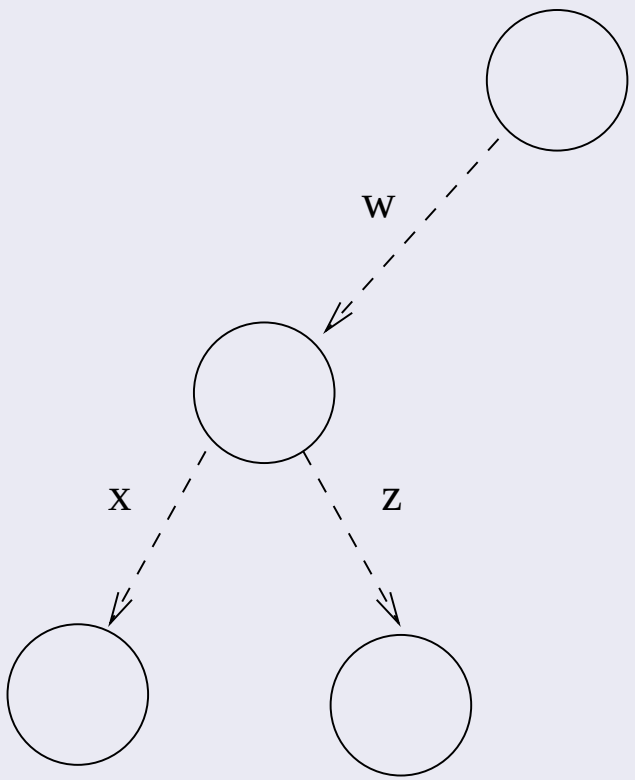
Règle 1



Les 3 règles

L'algorithme d'Ukkonen est basé sur 3 règles formulées par Gusfield :

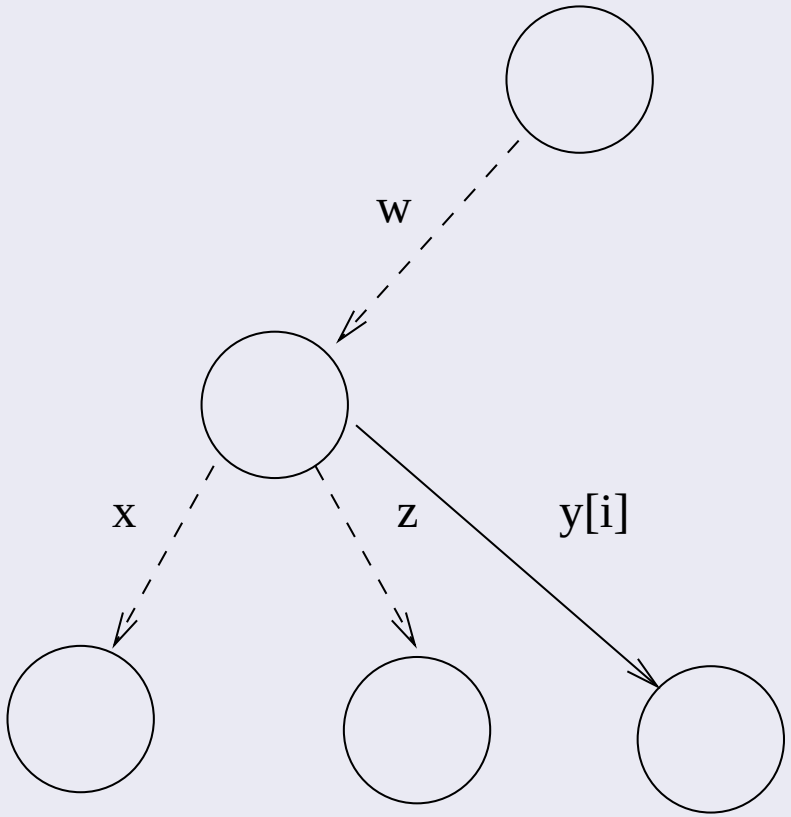
Règle 2 - cas A



Les 3 règles

L'algorithme d'Ukkonen est basé sur 3 règles formulées par Gusfield :

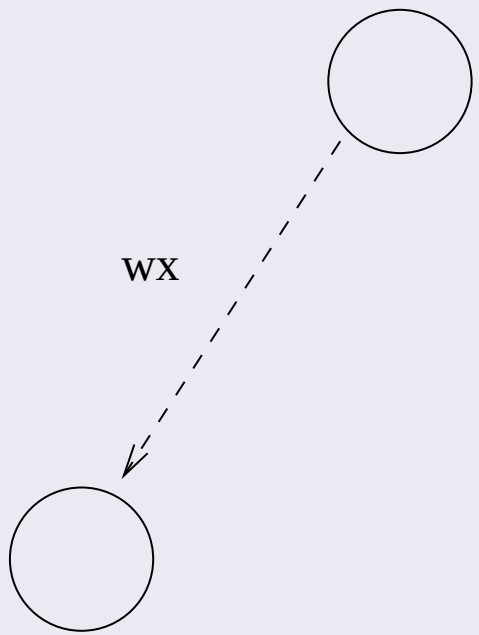
Règle 2 - cas A



Les 3 règles

L'algorithme d'Ukkonen est basé sur 3 règles formulées par Gusfield :

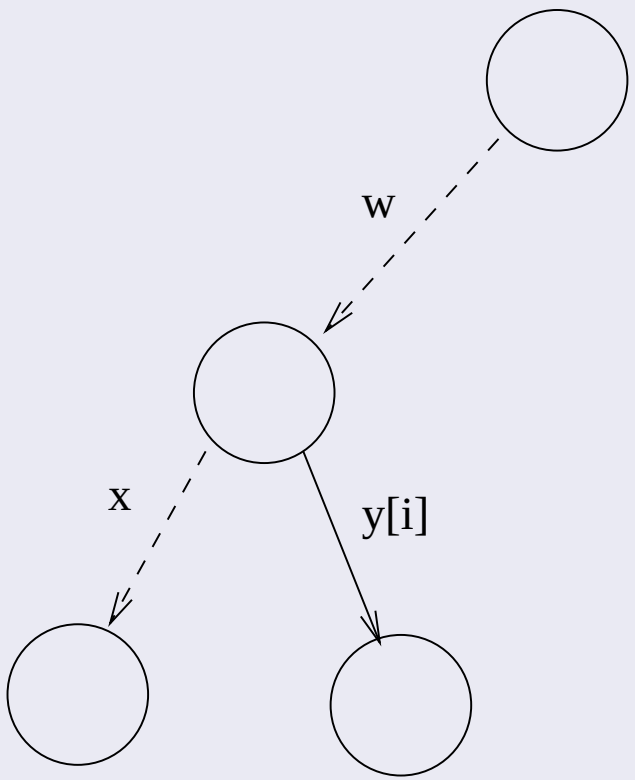
Règle 2 - cas B



Les 3 règles

L'algorithme d'Ukkonen est basé sur 3 règles formulées par Gusfield :

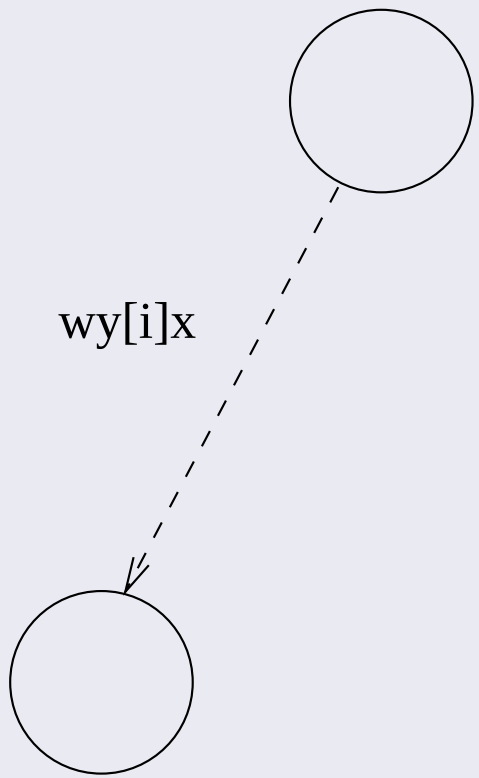
Règle 2 - cas B



Les 3 règles

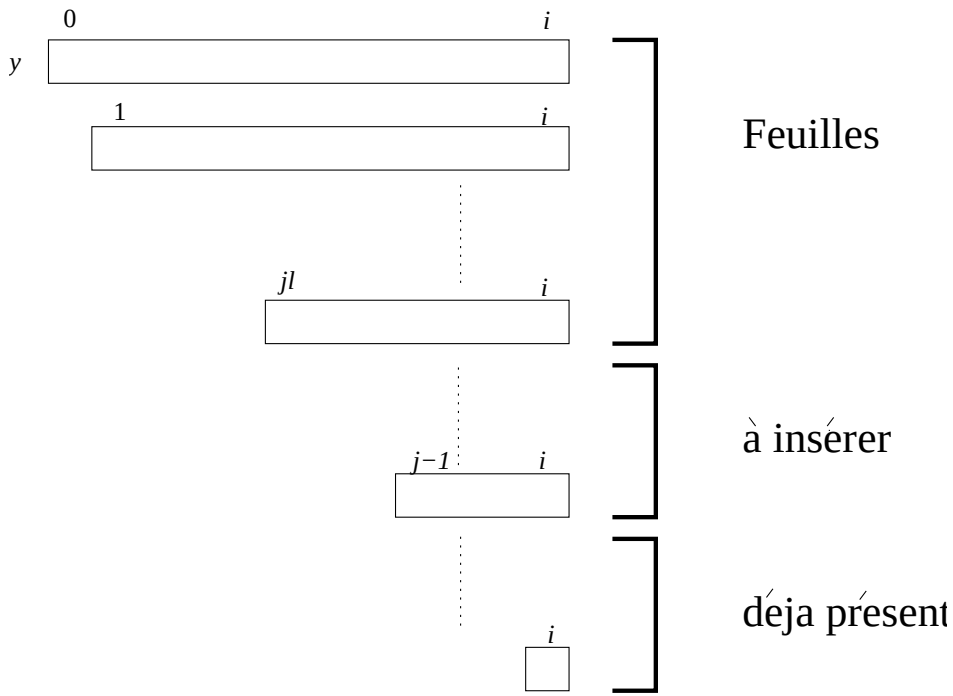
L'algorithme d'Ukkonen est basé sur 3 règles formulées par Gusfield :

Règle 3



Quelques propriétés

- la règle 1 ne nécessite aucun traitement,
- une phase i commence à l'extension $j_\ell + 1$, où j_ℓ est le numéro de la dernière feuille créée,
- une phase i s'achève à la première extension $j > j_\ell$ pour laquelle s'applique la règle 3.



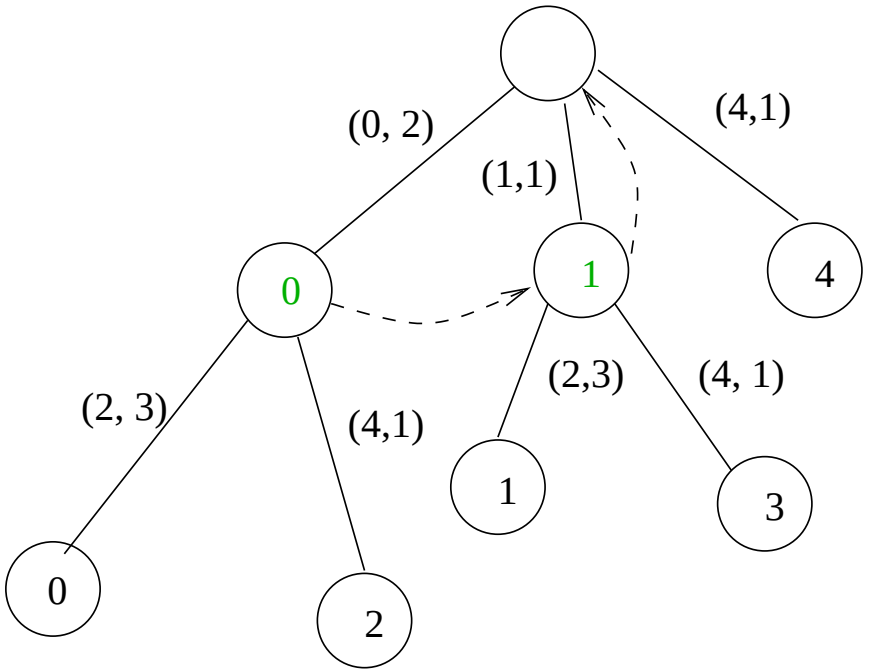
Introduction aux vecteurs de suffixes

Racine

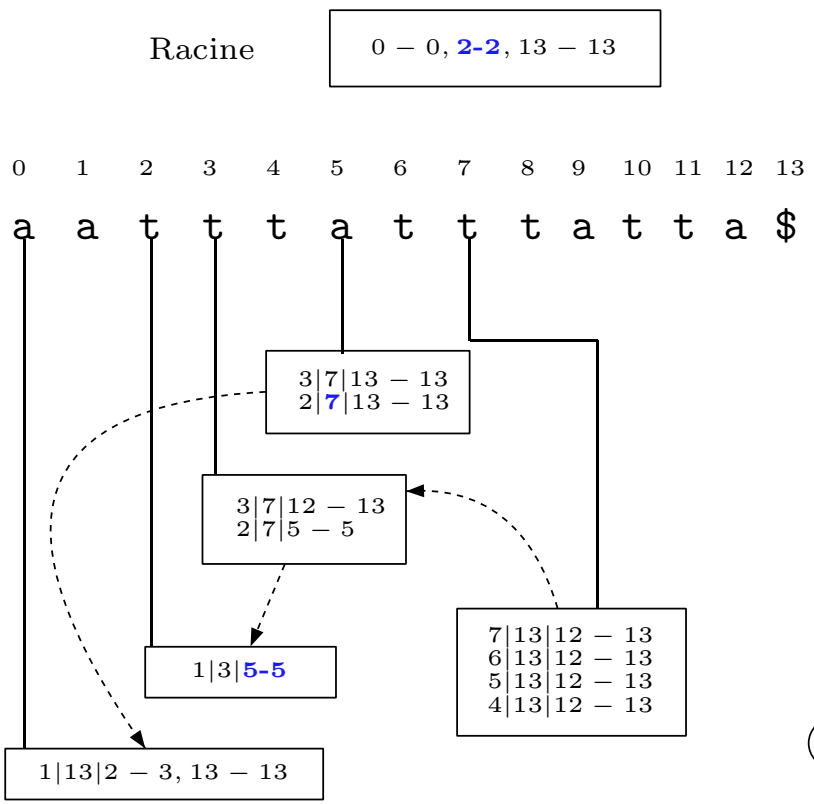
0 – 1,	1 – 1 ,	4 – 4

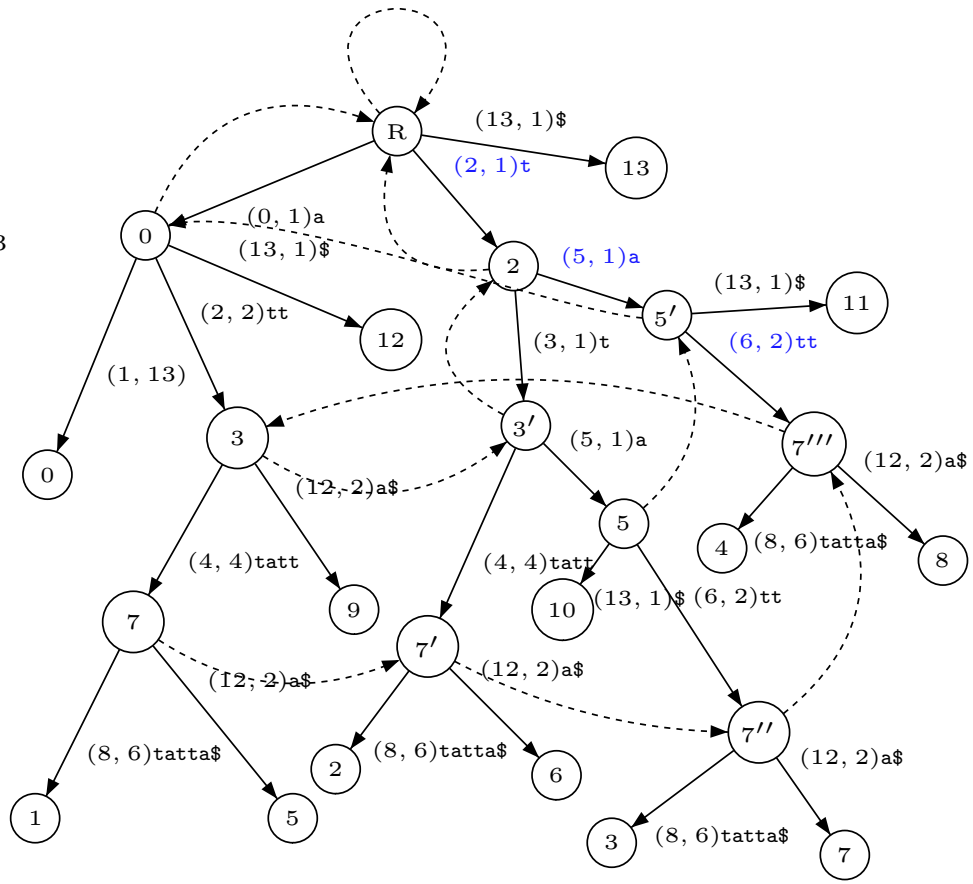
0	1	2	3	4
t	a	t	a	$

2	4	4 – 4
1	4	4 – 4

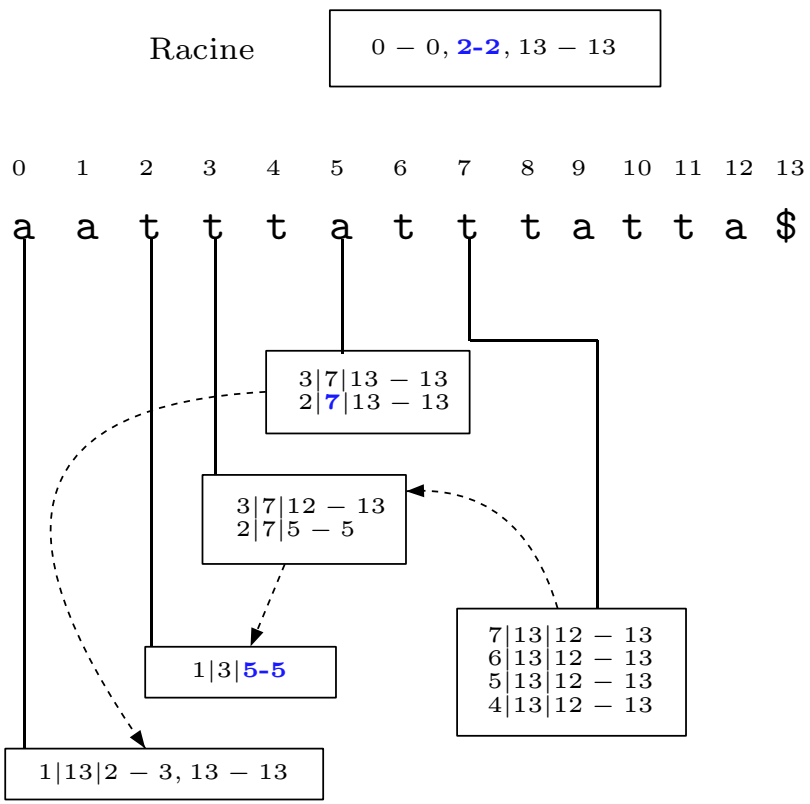


Introduction aux vecteurs de suffixes



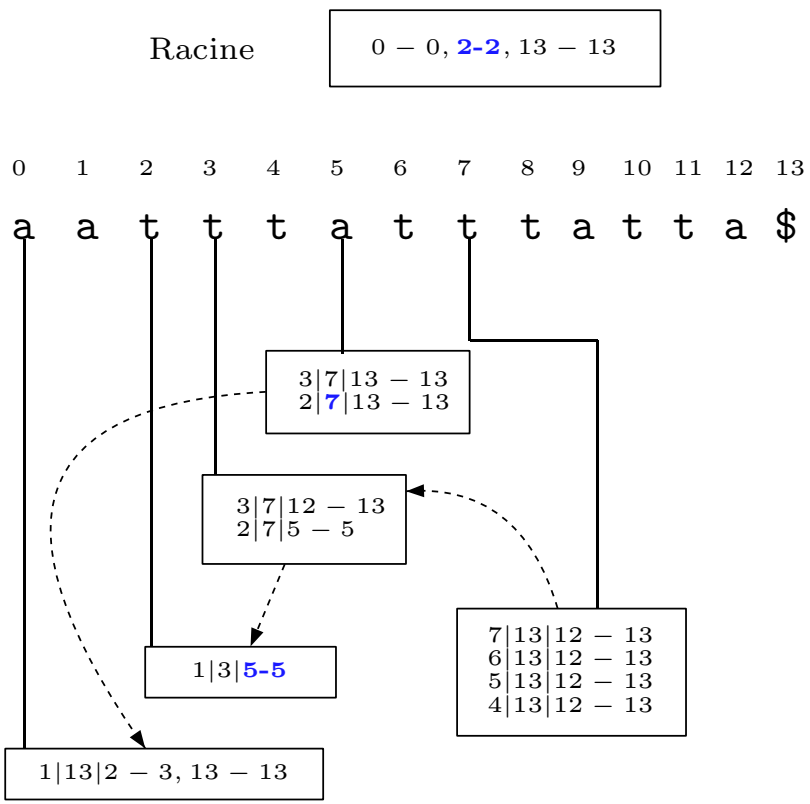


Introduction aux vecteurs de suffixes



- Structure de données alternative à l'arbre de suffixes
- mêmes informations dans un espace moindre
- introduit par K. Monostori en 2001

Introduction aux vecteurs de suffixes



Définition

Une succession de boîtes dont les lignes contiennent :

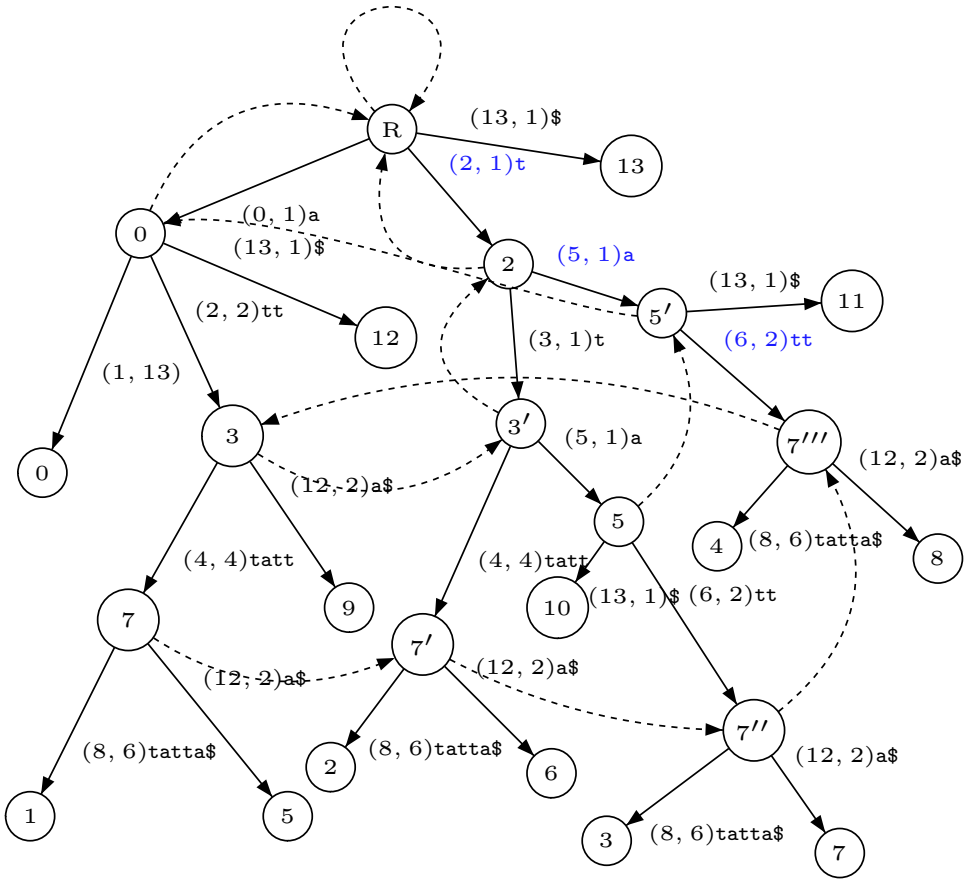
- la profondeur du nœud ;
- le chemin naturel ;
- la liste des transitions.

La racine est une boîte particulière.

Notations

- B_j : boîte à la position j dans y,
- Le chemin naturel, *cn*, d'une ligne dans B_j est la position de fin de la transition commençant par $y[j+1]$.

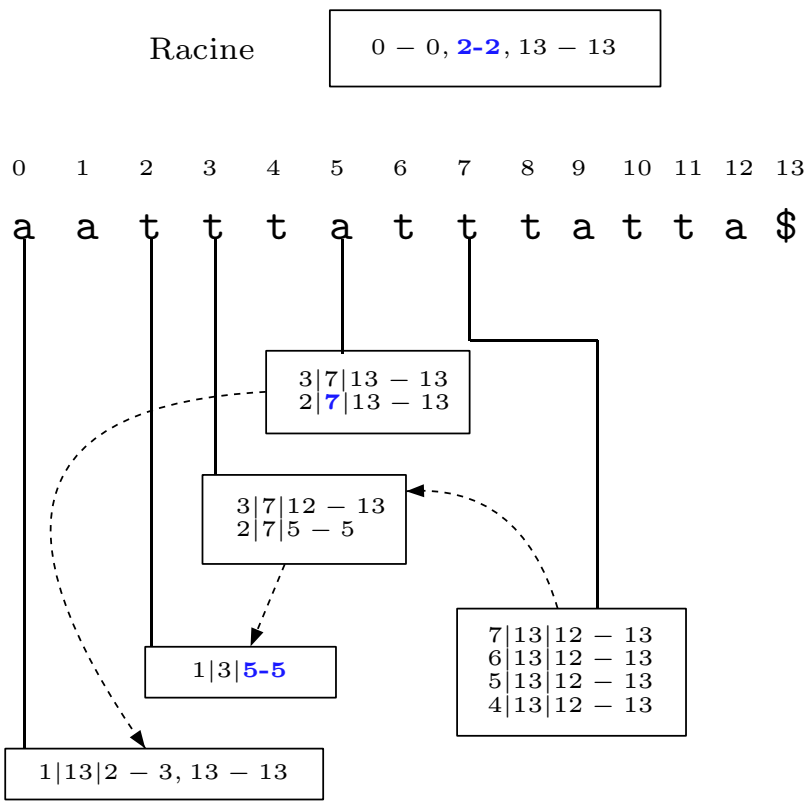
Introduction aux vecteurs de suffixes



Exemple

tatt est - il un facteur de y ?
La racine contient la transition $(2, 1)$ débutant par **t** arrivant à B_2.
La transition $(5, 1)$ par **a** mène à B_5.
La transition $(6, 2)$ par **tt** donne une occurrence de `tatt`.

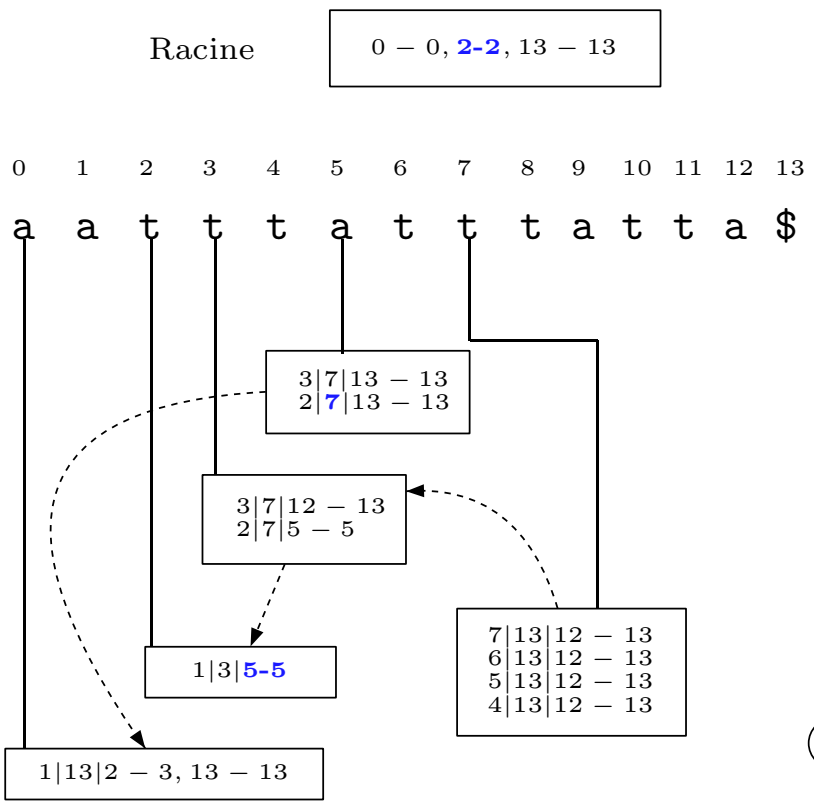
Introduction aux vecteurs de suffixes

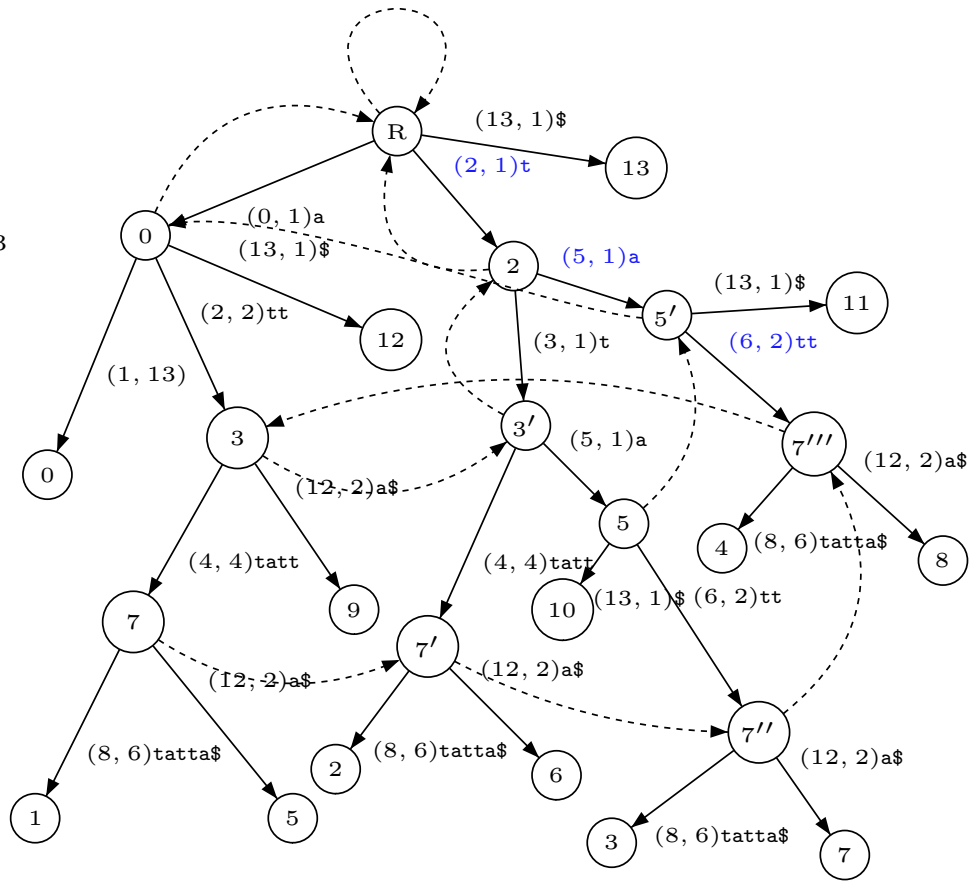


Exemple

tatt est - il un facteur de y ?
La racine contient la transition $(2, 2)$ débutant par **t** arrivant à B_2.
La transition $(5, 5)$ par **a** mène à B_5.
Le chemin naturel commence par **tt**.

Équivalence entre un vecteur et un arbre





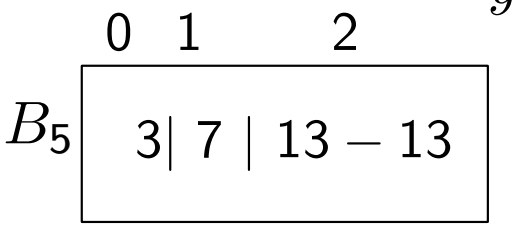
Exemple

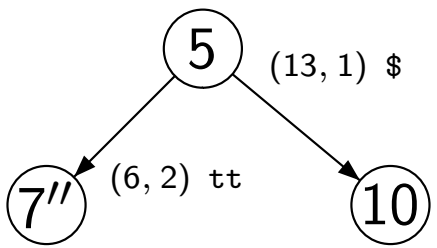
La ligne 0 de la boîte en position 5 représente le facteur $y[5-3+1..5] = \texttt{tta}$, i.e. le nœud 5 de $\mathcal{A}(y)$. On a donc :

- la transition allant de `tta` au nœud $p = \texttt{tta}y[6..7] = \texttt{ttatt}$ (car $B_5[0,1] = 7$) ;
- la transition étiquetée $(13, 1)$ allant vers une feuille (car $B_5[0,2] = (13,13)$).

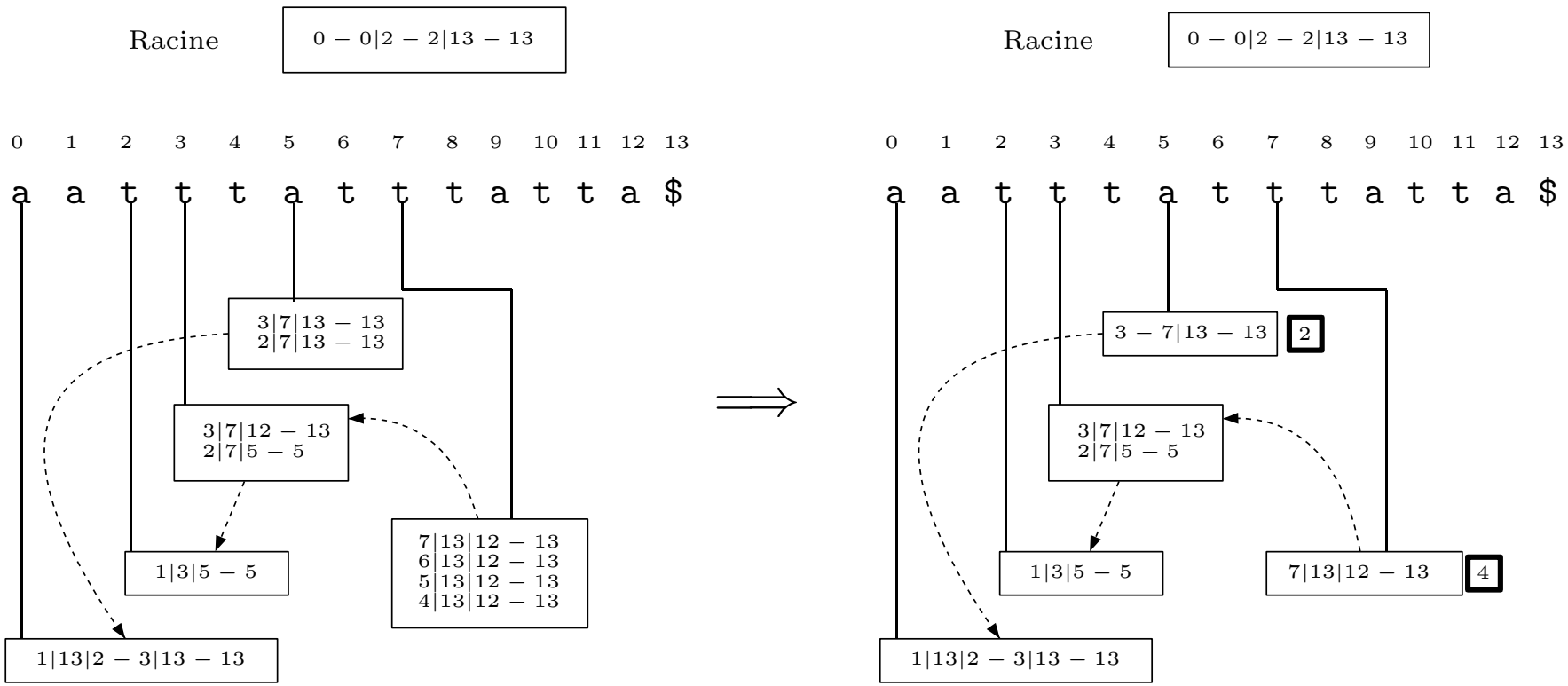
$y = \texttt{aatttatttatta\$}$

$u = \texttt{tta}$





Compaction de $\mathcal{V}$(aatttatttatta$)



Compaction d'un vecteur

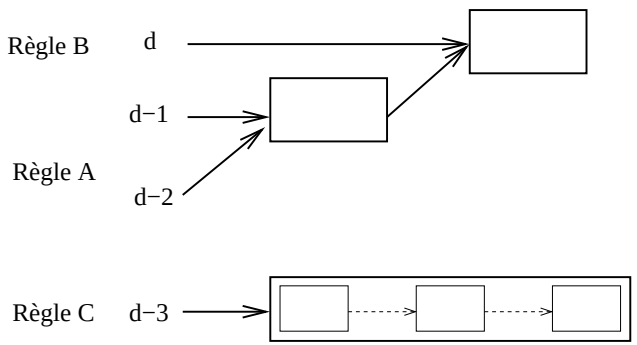
Définition

On appelle *groupe de nœuds* un ensemble de nœuds appartenant à une même boîte et qui ont exactement les mêmes transitions.

Vecteurs compacts de suffixes

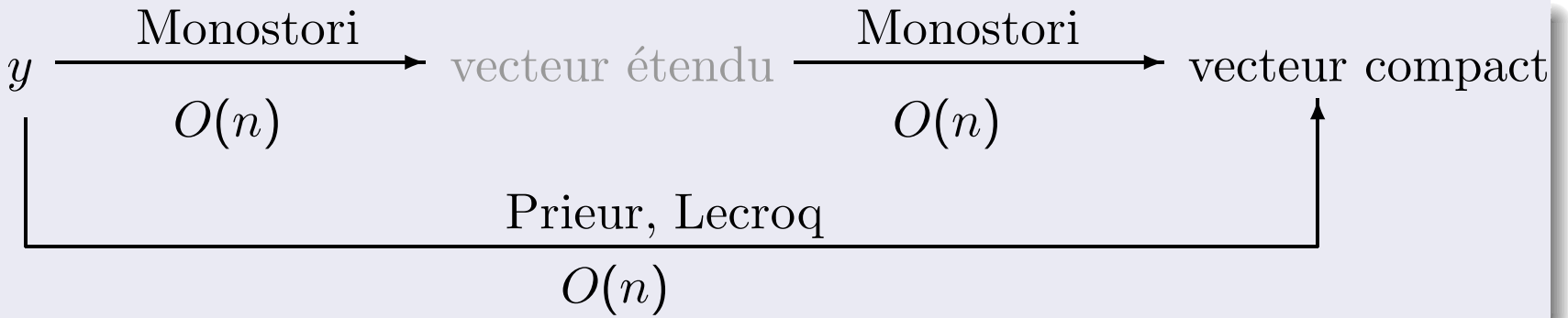
3 règles de compaction d'une boîte :

- **Règle A** le nœud de profondeur $d-2$ a les mêmes transitions que le nœud de profondeur $d-1$,
- **Règle B** le nœud de profondeur $d-1$ a les mêmes transitions que le nœud de profondeur d et des transitions supplémentaires,
- **Règle C** le nœud de profondeur $d-3$ a des transitions différentes du nœud de profondeur $d-2$.



$$y \xrightarrow[O(n)]{\text{Monostori}} \text{Vecteur étendu} \xrightarrow[O(n)]{\text{Monostori}} \text{Vecteur compact}$$

Construction directe d'un vecteur compact

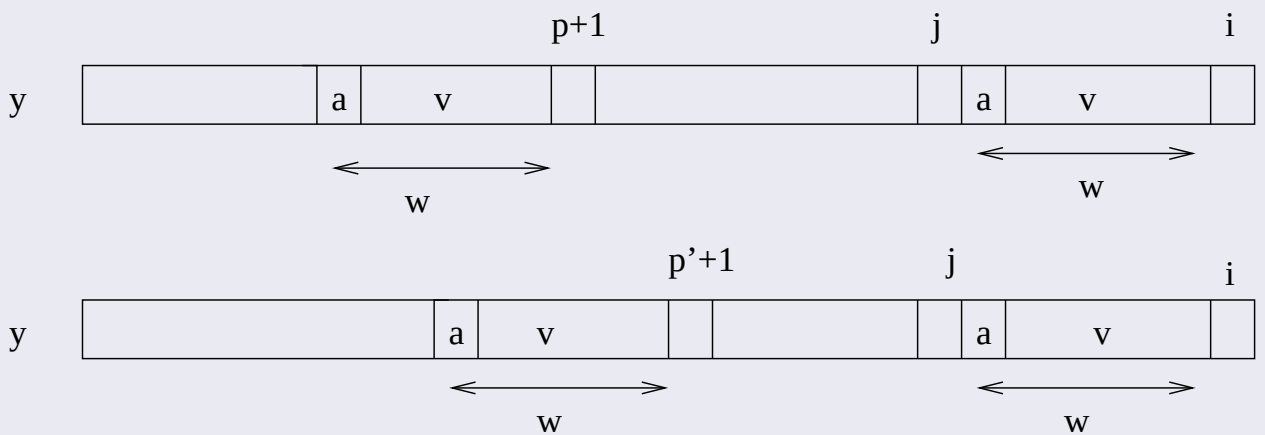


Construction plus rapide et plus économique en espace.

Construction directe d'un vecteur compact

Proposition 1

Lorsqu'une transition est ajoutée au nœud w de profondeur d dans une boite B_p, cette transition devra être ajoutée à tous les nœuds de B_p de profondeur inférieure à d appartenant au même groupe de nœuds que w.



Corollaire 1

Lorsqu'une transition est ajoutée au nœud de plus grande profondeur d'une boîte réduite B_p, cette transition devra être ajoutée à tous les nœuds de B_p. La boîte B_p restera réduite.

Résultats

- Monostori : 1 action par extension, autant d'extensions que l'algorithme d'Ukkonen ;
- construction directe du vecteur compact : possibilité de faire plusieurs actions dans une extension et donc de sauter des extensions.

Définitions

Soient x_1 et x_2 deux mots de A^*, on définit la relation d'équivalence $\mathcal{R}_y$ par $x_1 \mathcal{R}_y x_2 \iff \mathit{Posfin}_y(x_1) = \mathit{Posfin}_y(x_2)$, où $\mathit{Posfin}_y(x) = \{k \mid y = zxy[k+1..n-1]\}$.

La classe d'équivalence $\mathit{Cl}_{\mathcal{R}_y}(x)$ est définie par :
$\mathit{Cl}_{\mathcal{R}_y}(x) = \{x' \mid x' \mathcal{R}_y x\}$

Exemple

$y = \texttt{aatttatttatta\$}$
$\mathit{Posfin}_y(\texttt{tta}) = \{5, 9, 12\}$ $\mathit{Posfin}_y(\texttt{ta}) = \{5, 9, 12\}$
donc $\texttt{ta} \mathcal{R}_y \texttt{tta}$ et $\mathit{Cl}_{\mathcal{R}_y}(\texttt{tta}) = \{\texttt{tta}, \texttt{ta}\}$

Définition

Une répétition maximale dans une séquence est un facteur u tel qu'il existe au moins 2 occurrences : a_1ub_1 et a_2ub_2 avec $a_1 \neq a_2$, $b_1 \neq b_2$ et $a_1, a_2, b_1, b_2 \in A$.

Exemple

$y =$aatttatttatta$

tta est une répétition maximale aux positions 5 et 12.

Définition

Une répétition maximale dans une séquence est un facteur u tel qu'il existe au moins 2 occurrences : a_1ub_1 et a_2ub_2 avec $a_1 \neq a_2$, $b_1 \neq b_2$ et $a_1, a_2, b_1, b_2 \in A$.

Exemple

$y =$aatttatttatta$

tta est une répétition maximale aux positions 5 et 12.

Théorème (Raffinot, 2001)

Un facteur est une répétition maximale si et seulement si il est le plus long mot d'une classe d'équivalence.

Liens avec les vecteurs de suffixes

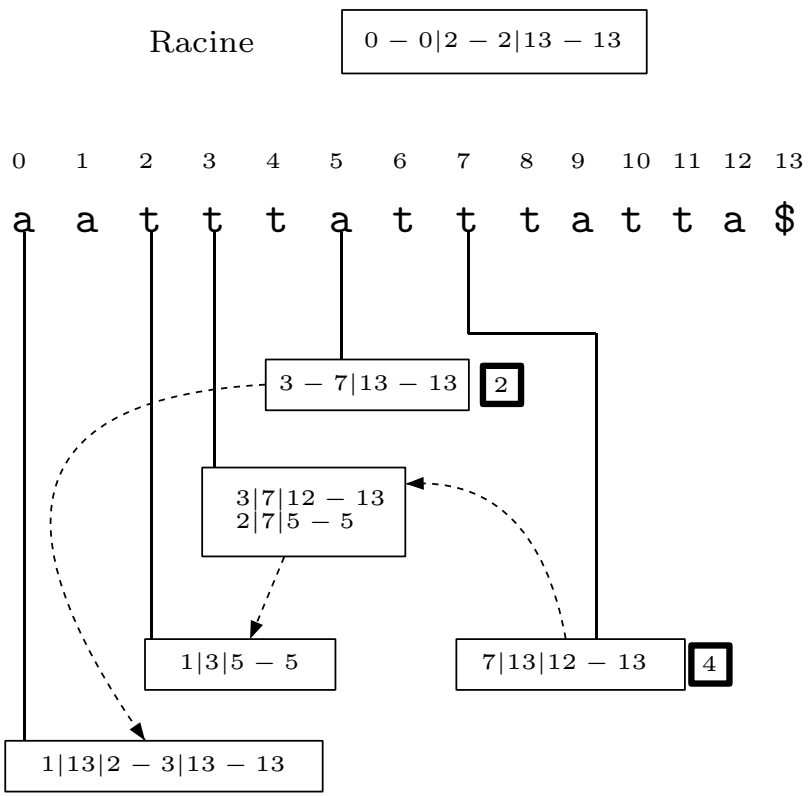
Proposition 2

Les facteurs appartenant à une même classe $Cl_{\mathcal{R}_y}(x)$ sont représentés dans la même boîte du vecteur de y. C'est la boîte à la position p tel que $p = min\{k \mid k \in Posfin_y(x)\}$.

Proposition 3

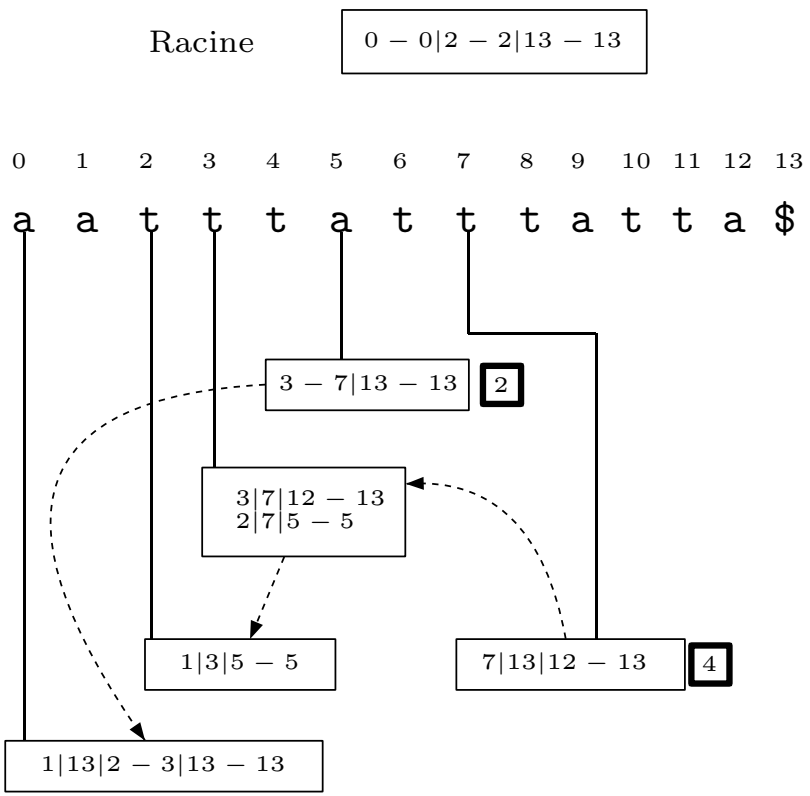
Le nœud le plus long de chaque groupe de nœuds représente une répétition maximale.

Dans une boîte réduite, le nœud de plus grande profondeur est le seul à représenter une répétition maximale.



Exemple

Les boîtes 0, 2, 5 et 7 sont réduites : `a, t, tta, atttatt` sont des répétitions maximales.
La boîte B_3 est étendue, les 2 lignes ont des transitions différentes : `att, tt` sont des répétitions maximales.



Méthode

Visite de chaque boîte, le plus grand nœud de chacune est une répétition maximale. Si la boîte est réduite c'est le seul, sinon chaque ligne h telle que ses transitions diffèrent de celles de la ligne $h-1$ est une répétition maximale.

Conclusion

- équivalence entre un arbre de suffixes et un vecteur de suffixes ;
- gain en temps et en espace avec la construction directe du vecteur de suffixes compact ;
- méthode linéaire pour calculer les répétitions maximales avec un vecteur de suffixes compact.

Perspectives

- optimiser l'implantation de la structure de vecteur de suffixes compact afin de traiter de plus longues séquences ;
- utiliser les vecteurs de suffixes pour la détection de répétitions particulières (éléments transposables, ...).

Arbres de suffixes


M. Farach.
Optimal suffix tree construction with large alphabets.
In *Proceedings of the 38th IEEE FOCS* , pages 137–143, Miami Beach, FL, 1997.


D. Gusfield.
Algorithms on Strings, Trees and Sequences : Computer Science and Computational Biology.
Cambridge University Press, Cambridge, 1997.


S. Kurtz.
Reducing the space requirements of suffix trees.
Software Practice & Experience, 29(13) :1149–1171, 1999.


E. M. McCreight.
A space-economical suffix tree construction algorithm.
Journal of Algorithms, 23(2) :262–272, 1976.


E. Ukkonen.
On-line construction of suffix trees.
Algorithmica, 14(3) :249–260, 1995.

Vecteurs de suffixes


K. Monostori.
Efficient Computational Approach to Identifying Overlapping Documents in Large Digital Collections.
PhD thesis, Monash University, 2002.


K. Monostori, A. Zaslavsky, and H. Schmidt.
Suffix vector : Space-and-time-efficient alternative to suffix trees, 2002.
In *CRPITS '02 : Proceedings of the 25th ACSC* , volume 4, pages 157–166, Melbourne, 2002.


K. Monostori, A. Zaslavsky, et I. Vajk.
Suffix vector : A space-efficient suffix tree representation, 2001.
In *Proceedings of the 12th ISAC*, volume 2223 of *LNCS*, pages 707–718, Christchurch, New Zealand, 2001. Springer Verlag.


É. Prieur, et T. Lecroq.
From suffix trees to suffix vectors, 2005.
In *Proceedings of PSC'05*, Prague, Czech Republic, 2005.

Répétitions maximales


D. Gusfield.
Algorithms on Strings, Trees and Sequences : Computer Science and Computational Biology.
Cambridge University Press, Cambridge, 1997.


M. Raffinot.
On maximal repeats in strings.
In *Information Processing Letters*, 80(6) :165–169, 2001.