Algorithmes de texte

Cours L3 Algorithmique et programmation

Pierre Senellart, Antoine Amarilli

23 novembre 2017

Plan

Trouver les occurrences d'une sous-chaîne dans une chaîne

- Algorithme naïf :
  `<MINTED>`
- En général, similaire à l'implémentation native du langage (`strstr`, `string::find`), fortement optimisée

Trouver les occurrences d'une sous-chaîne dans une chaîne

- Algorithme naïf :
  `<MINTED>`
- En général, similaire à l'implémentation native du langage (`strstr`, `string::find`), fortement optimisée
- Complexité $O(|s| \times |p|)$
- Peut-on mieux faire ?

Knuth–Morris–Pratt : idée

- Motif p, chaîne s
- Pour chaque préfixe p' de p, maintenir la taille du préfixe maximal de p qui est un suffixe strict de p'
- $p =$“abcababcabd”
  00012123450
- Tableau constructible en temps linéaire en le motif p

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T													
i													
cnd													

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1											
i													
cnd													

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1											
i			↑										
cnd		↑											

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0										
i			↑										
cnd		↑											

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0										
i			↑										
cnd	↑												

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0										
i			↑										
cnd		↑											

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0										
i				↑									
cnd		↑											

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0									
i				↑									
cnd		↑											

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0									
i				↑									
cnd	↑												

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0									
i				↑									
cnd		↑											

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0									
i					↑								
cnd		↑											

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0								
i					↑								
cnd		↑											

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0								
i					↑								
cnd			↑										

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0								
i						↑							
cnd			↑										

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1							
i						↑							
cnd			↑										

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1							
i						↑							
cnd				↑									

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1							
i							↑						
cnd				↑									

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2						
i							↑						
cnd				↑									

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2						
i							↑						
cnd		↑											

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2						
i							↑						
cnd			↑										

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2						
i								↑					
cnd			↑										

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2	1					
i								↑					
cnd			↑										

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2	1					
i								↑					
cnd				↑									

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2	1					
i									↑				
cnd				↑									

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2	1	2				
i									↑				
cnd				↑									

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2	1	2				
i									↑				
cnd					↑								

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2	1	2				
i										↑			
cnd					↑								

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2	1	2	3			
i										↑			
cnd					↑								

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2	1	2	3			
i										↑			
cnd						↑							

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2	1	2	3			
i											↑		
cnd						↑							

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2	1	2	3	4		
i											↑		
cnd						↑							

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2	1	2	3	4		
i											↑		
cnd							↑						

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2	1	2	3	4		
i												↑	
cnd							↑						

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2	1	2	3	4	5	
i												↑	
cnd							↑						

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2	1	2	3	4	5	
i												↑	
cnd				↑									

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2	1	2	3	4	5	
i												↑	
cnd		↑											

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2	1	2	3	4	5	
i												↑	
cnd	↑												

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2	1	2	3	4	5	
i												↑	
cnd		↑											

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2	1	2	3	4	5	
i													↑
cnd		↑											

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2	1	2	3	4	5	0
i													↑
cnd		↑											

Knuth–Morris–Pratt : calcul du tableau

<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2	1	2	3	4	5	0
i													↑
cnd	↑												

Knuth–Morris–Pratt : calcul du tableau

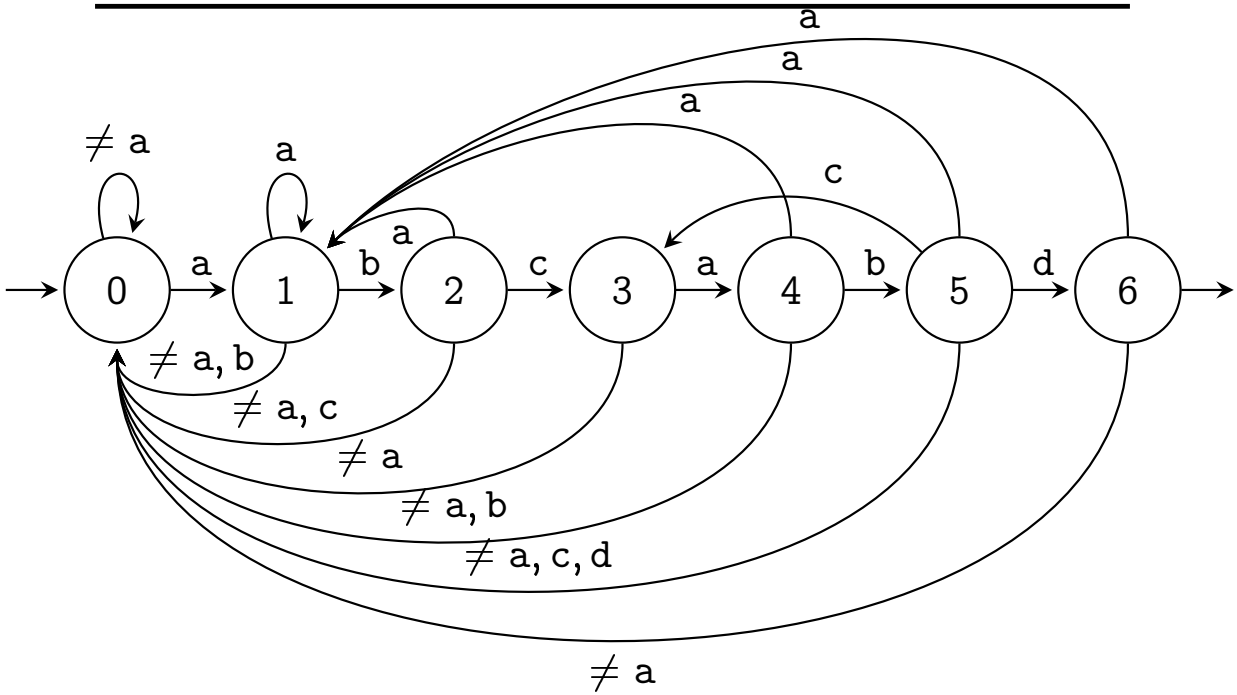
<MINTED>

	-1	0	1	2	3	4	5	6	7	8	9	10	11
p		a	b	c	a	b	a	b	c	a	b	d	
T		-1	0	0	0	1	2	1	2	3	4	5	0
i													↑
cnd		↑											

Knuth–Morris–Pratt : interprétation comme automate

T peut être vu comme l'automate déterministe des mots ayant pour suffixe p. Par exemple, pour $p =$ abcabd :

	0	**1**	**2**	**3**	**4**	**5**	**6**
p	a	b	c	a	b	d	
T	−1	0	0	0	1	2	0



Knuth–Morris–Pratt: recherche

<MINTED>

Plan

Recherche d'un mot parmi un ensemble de mots

- Dictionnaire D de n mots de longueur $\leqslant \ell$
- Pour rechercher si un mot est dans cet ensemble :
  - Tableau (`vector`) ou liste chaînée (`forward_list`) : $O(n \times \ell)$
  - Arbre binaire équilibré (`set`) : $O(\log n \times \ell)$
  - Table de hachage (`unordered_set`): $O(\ell)$
    mais $O(n \times \ell)$ dans le pire cas (collisions)

Recherche d'un mot parmi un ensemble de mots

- Dictionnaire D de n mots de longueur $\leqslant \ell$
- Pour rechercher si un mot est dans cet ensemble :
  - Tableau (`vector`) ou liste chaînée (`forward_list`) : $O(n \times \ell)$
  - Arbre binaire équilibré (`set`) : $O(\log n \times \ell)$
  - Table de hachage (`unordered_set`): $O(\ell)$ mais $O(n \times \ell)$ dans le pire cas (collisions)
  - Filtre de Bloom : $O(\ell)$ et occupation mémoire indépendante de ℓ, mais possibilité de faux positifs

Recherche d'un mot parmi un ensemble de mots

- Dictionnaire D de n mots de longueur $\leqslant \ell$
- Pour rechercher si un mot est dans cet ensemble :
  - Tableau (`vector`) ou liste chaînée (`forward_list`) : $O(n \times \ell)$
  - Arbre binaire équilibré (`set`) : $O(\log n \times \ell)$
  - Table de hachage (`unordered_set`): $O(\ell)$ mais $O(n \times \ell)$ dans le pire cas (collisions)
  - Filtre de Bloom : $O(\ell)$ et occupation mémoire indépendante de ℓ, mais possibilité de faux positifs
  - Trie (ou arbre préfixe) : $O(\ell)$ et occupation mémoire plus faible que structures de données classiques

Filtre de Bloom

Au tableau.

Filtre de Bloom

Au tableau.

Antisèche :

- tableau de taille m, n éléments, k fonctions de hachage
- Probabilité d'erreur:

$$\left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k \sim \left(1 - \mathrm{e}^{-kn/m}\right)^k$$

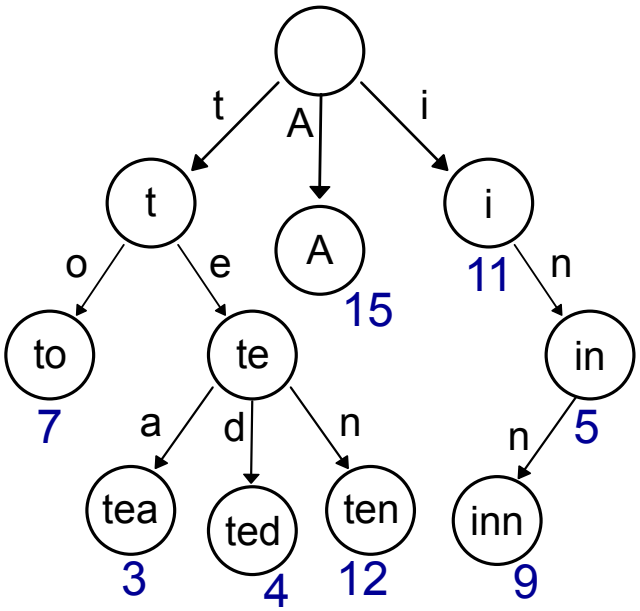
- k optimal:

$$\frac{m \ln 2}{n}$$

- $\frac{m}{n}$ optimal:

$$-\frac{\log_2 p}{\ln 2}$$

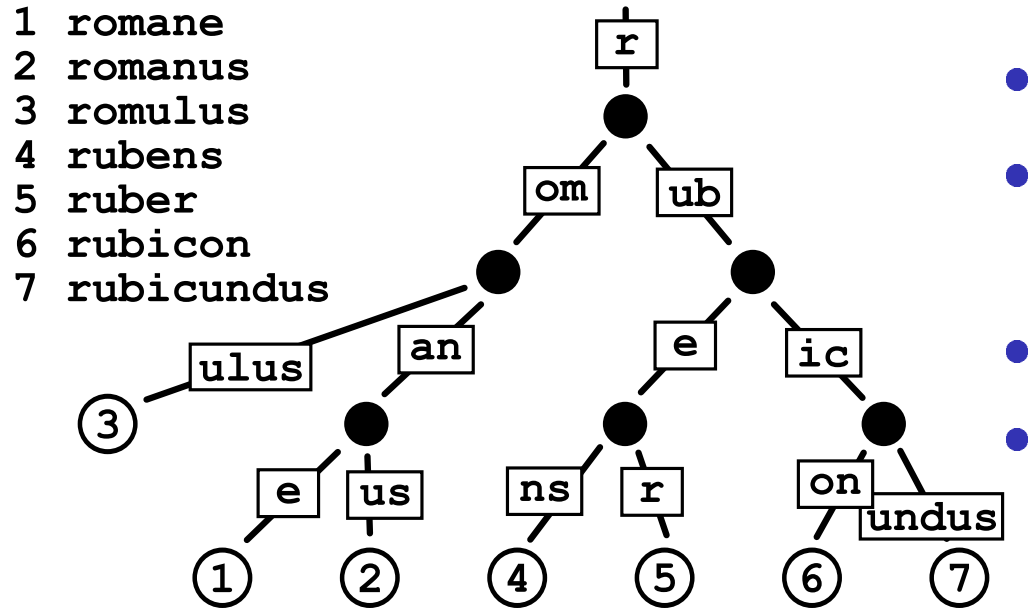
Trie (arbre préfixe)



Trie_example.svg, Domaine public, Chris-martin, Wikimedia Commons

- Arbre des préfixes des mots de D
- Arêtes étiquetées par des lettres
- On indique sur chaque nud l'id dans D du chemin qui y mène
- Permet de trouver les continuations du mot d'entrée m dans D, i.e., les mots de D dont m est préfixe
- L'ordre lexicographique est préservé
- Bonne occupation mémoire si l'alphabet est petit

Arbre radix (trie Patricia)



Patricia trie.svg, CC-BY 2.5, Claudio Rocchini,

Wikimedia Commons

- Raffinement des tries
- Fusionne les enfants uniques avec leur parent
- Espace mémoire réduit
- Branchement :
  - par bit (entiers),
  - par lettre,
  - par groupe de bits (radix : taille du groupe)

Plan

Trouver les occurrences d'un ensemble de sous-chaînes dans une chaîne

- Dictionnaire D de taille n contenant des mots de taille k
- Chaîne de taille ℓ

Trouver les occurrences d'un ensemble de sous-chaînes dans une chaîne

- Dictionnaire D de taille n contenant des mots de taille k
- Chaîne de taille ℓ
- Applications répétées de Knuth–Morris–Pratt ?
  $O(n \times (k + \ell))$

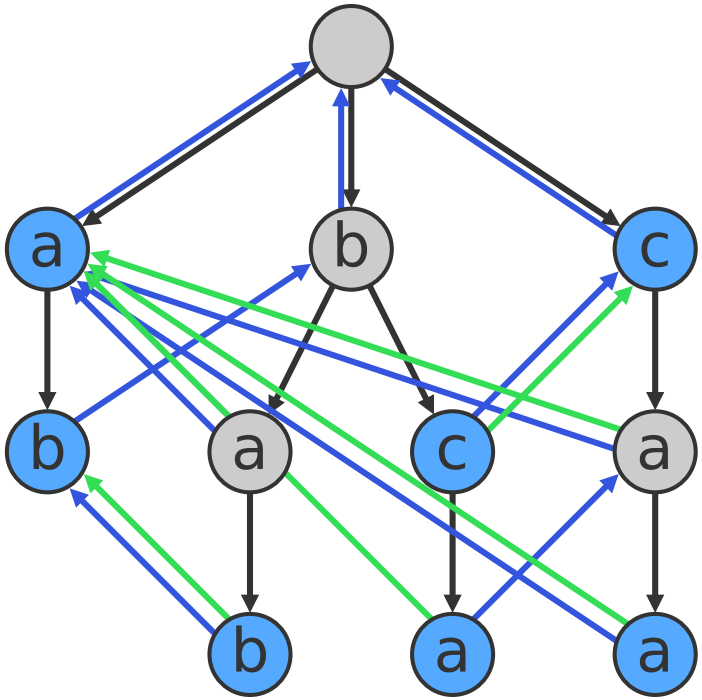
Trouver les occurrences d'un ensemble de sous-chaînes dans une chaîne

- Dictionnaire D de taille n contenant des mots de taille k
- Chaîne de taille ℓ
- Applications répétées de Knuth–Morris–Pratt ?
  $O(n \times (k + \ell))$
- On peut généraliser Knuth–Morris–Pratt à un trie arbitraire (et non une séquence de caractères) :
  $O(n \times k + \ell + m)$ où m est le nombre total d'occurrences (taille du résultat), au plus $k \times \ell$ mais possiblement plus faible

Trouver les occurrences d'un ensemble de sous-chaînes dans une chaîne

- Dictionnaire D de taille n contenant des mots de taille k
- Chaîne de taille ℓ
- Applications répétées de Knuth–Morris–Pratt ?
  $O(n \times (k + \ell))$
- On peut généraliser Knuth–Morris–Pratt à un trie arbitraire (et non une séquence de caractères) :
  $O(n \times k + \ell + m)$ où m est le nombre total d'occurrences (taille du résultat), au plus $k \times \ell$ mais possiblement plus faible
- Approche alternative : indexer la chaîne plutôt qu'indexer les sous-chaînes $\Rightarrow$ arbre des suffixes, même complexité

Aho–Corasick

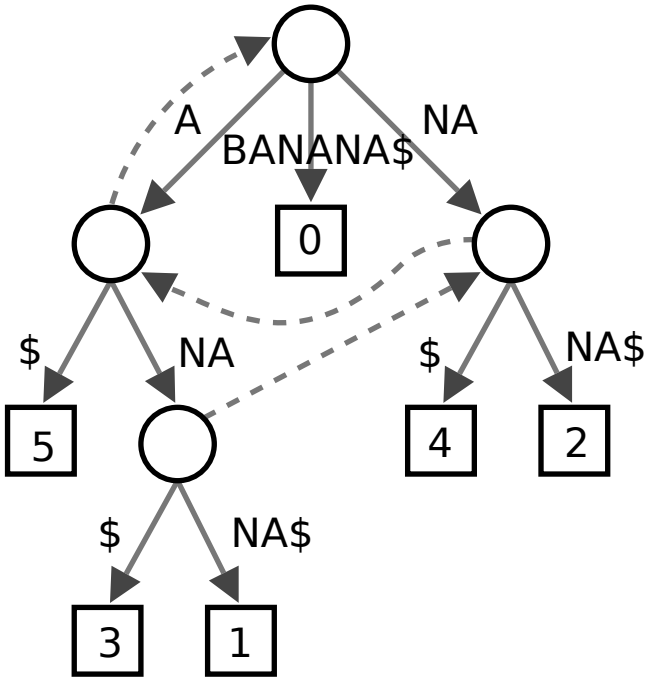


Ahocorasick.svg, CC-BY-SA 3.0,
Dllu, Wikimedia Commons

- Construire un trie du dictionnaire (liens en noir, mots sur fond bleu)
- Ajouter des pointeurs (en bleu) vers le plus grand suffixe strict dans le trie
- Ajouter des raccourcis (en vert) pour les mots du dictionnaire
- Exemple : abccab donne :
  - a (parcours normal),
  - ab (parcours normal),
  - bc (suivi du pointeur bleu) puis c (clôture par le pointeur vert),
  - c (suivi du pointeur bleu),
  - ca (suivi du pointeur bleu) puis a (clôture par le pointeur vert),
  - ab (suivi du pointeur bleu)

Dictionnaire : a, ab, bab, bc, bca, c, caa.

Arbre des suffixes



Suffix tree BANANA.svg,
Domaine public,
Maciej Jaros and Nils Grimsmo,
Wikimedia Commons

- Trie Patricia des suffixes d'un mot (ici, BANANA, suivi d'un délimiteur \$)
- Constructible en $O(\ell^2)$ de droite à gauche
- Constructible en $O\ell)$ de gauche à droite, comme les pointeurs d'Aho–Corasick (mais non trivial, algorithme de Ukkonen)
- Permet d'indexer une chaîne pour rechercher des sous-chaînes
- Nombreuses autres applications : p. ex., plus longue sous-chaîne commune

Plan

Expressions rationnelles

- Langage permettant de décrire des motifs à rechercher dans une chaîne de caractères
- Par exemple : `(a|b)*#(a|b)*(#(a|b|#)*)?`
- Généralise la recherche de sous-chaînes, de mots d'un dictionnaire, de préfixes, de suffixes, etc.
- Processeurs d'expressions rationnelles dans la bibliothèque standard de Python, Java, C++ 2011 (`std::regex`)...

Automates

- Expression rationnelle vers automate fini nondéterministe :
  - algorithme de Thompson : temps linéaire, transitions spontanées
  - algorithme de Glushkov : temps quadratique, moins d'états
- Reconnaître si une chaîne de longueur n est acceptée par un automate nondéterministe à m états est en $O(n \times m)$
- Un automate nondéterministe à m états peut être transformé en automate déterministe à $O(2^m)$ états en temps $O(2^m)$
- Reconnaître si une chaîne de longueur n est acceptée par un automate déterministe à m états est en $O(n)$