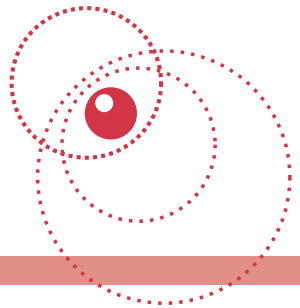


Entraînement au concours ACM-ICPC

Algorithmes de texte

8 juin 2015

Pierre Senellart

TELECOM ParisTech

Plan

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

- Algorithme naïf

```
// s est la chaîne, p le motif
for(int i=0,j;i<s.size()-p.size()+1;++i) {
  for(j=0;j<p.size() && s[i+j]==p[j];++j)
    ;

  if(j==p.size())
    cout << "Match à la position " << i << endl;
}
```

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

Knuth–Morris–Pratt : idée

- p motif, s chaîne
- Pour chaque préfixe p' de p, maintenir la taille du préfixe maximal de p qui est un suffixe strict de p'
- $p =$“`abcabd`”

  `00120`
- Tableau constructible en temps linéaire en le motif p
- $s =$“`abcababcabd`”

  `abcabd`

Knuth–Morris–Pratt : idée

- p motif, s chaîne
- Pour chaque préfixe p' de p, maintenir la taille du préfixe maximal de p qui est un suffixe strict de p'
- $p =$“abcabd”

  00120
- Tableau constructible en temps linéaire en le motif p
- $s =$“abcababcabd”

  abcabd

Knuth–Morris–Pratt : idée

- p motif, s chaîne
- Pour chaque préfixe p' de p, maintenir la taille du préfixe maximal de p qui est un suffixe strict de p'
- $p =$“abcabd”

  00120
- Tableau constructible en temps linéaire en le motif p
- $s =$“abcababcabd”

  abcabd

Knuth–Morris–Pratt : idée

- p motif, s chaîne
- Pour chaque préfixe p' de p, maintenir la taille du préfixe maximal de p qui est un suffixe strict de p'
- $p =$“abcabd”

  00120
- Tableau constructible en temps linéaire en le motif p
- $s =$“abcababcabd”

  abcabd

Knuth–Morris–Pratt : idée

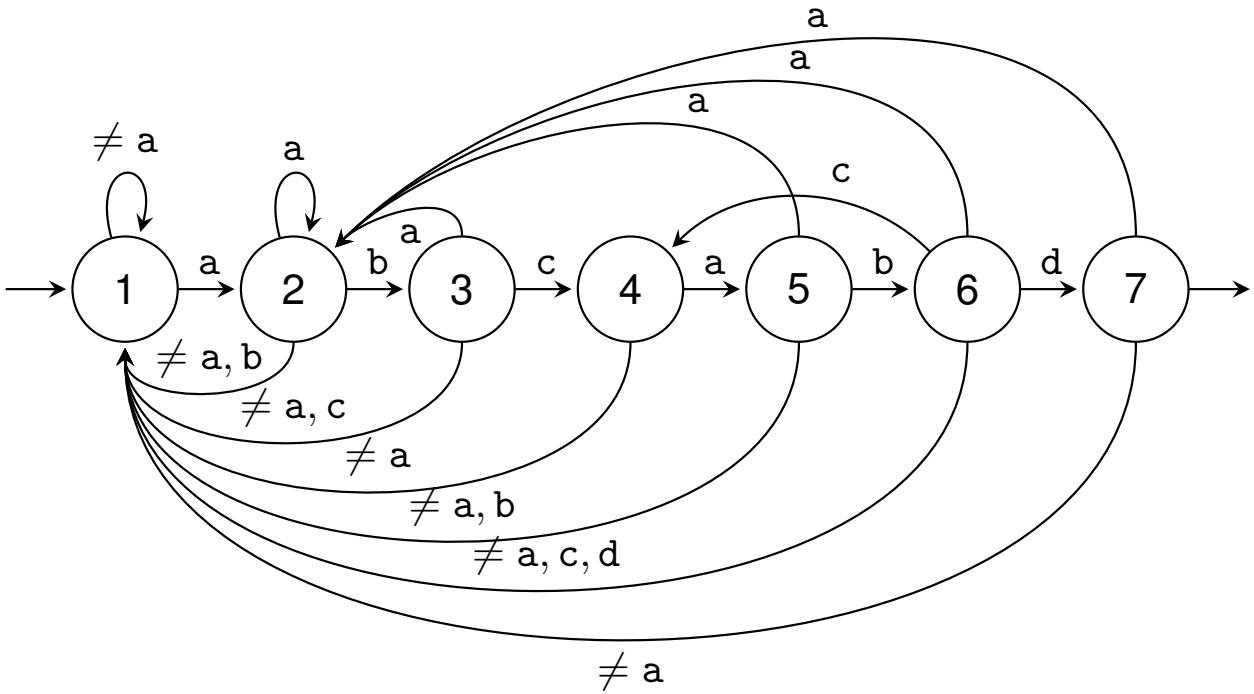
- p motif, s chaîne
- Pour chaque préfixe p' de p, maintenir la taille du préfixe maximal de p qui est un suffixe strict de p'
- $p =$“abcabd”

  00120
- Tableau constructible en temps linéaire en le motif p
- $s =$“abcababcabd”

  abcabd

Knuth–Morris–Pratt : interprétation comme automate

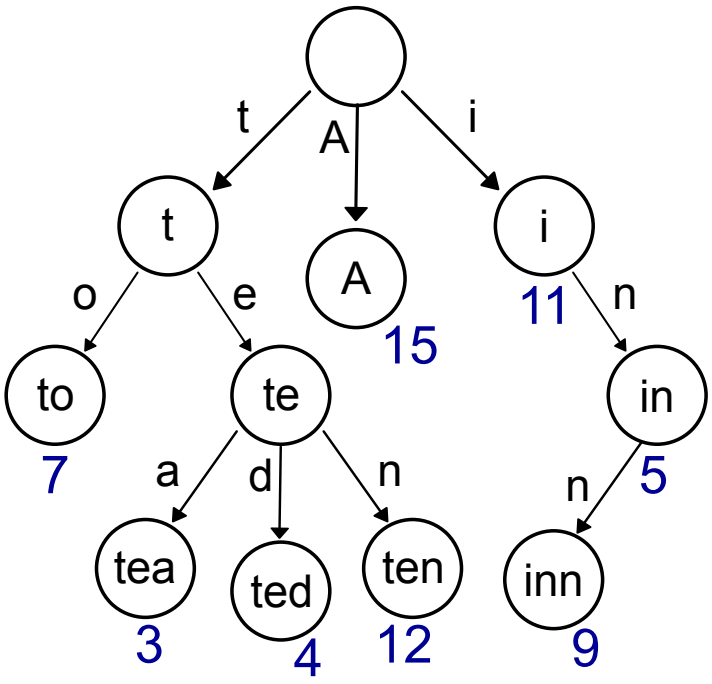
Peut-être vu comme l'automate déterministe des mots ayant pour suffixe p :



Recherche d'un mot parmi un ensemble de mots

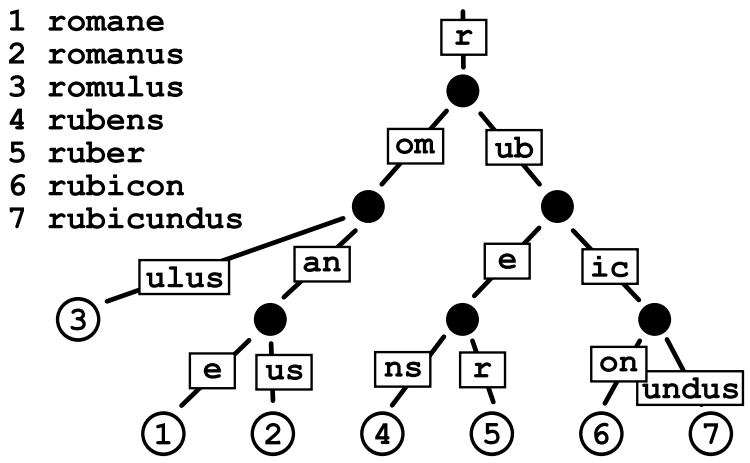
- Dictionnaire D de n mots
- Pour rechercher si un mot est dans cet ensemble :
  - Vecteur ou liste chaînée : $O(n)$
  - Arbre binaire équilibré : $O(\log n)$
  - Table de hachage : $O(1)$; possibilités de collision, $O(n)$ dans le pire des cas
  - Trie (ou arbre préfixe) : $O(1)$

Trie (arbre préfixe)



- Arbre de tous les préfixes des mots de *D*
- Arêtes étiquetées par les lettres de l'alphabet
- On indique sur chaque nœud de l'arbre si ce préfixe forme un des mots de *D*
- Permet de rechercher efficacement l'ensemble des continuations d'un mot *m* dans *D* (l'ensemble des mots de *D* dont *m* est préfixe)
- L'ordre lexicographique est préservé

Arbre radix (trie Patricia)



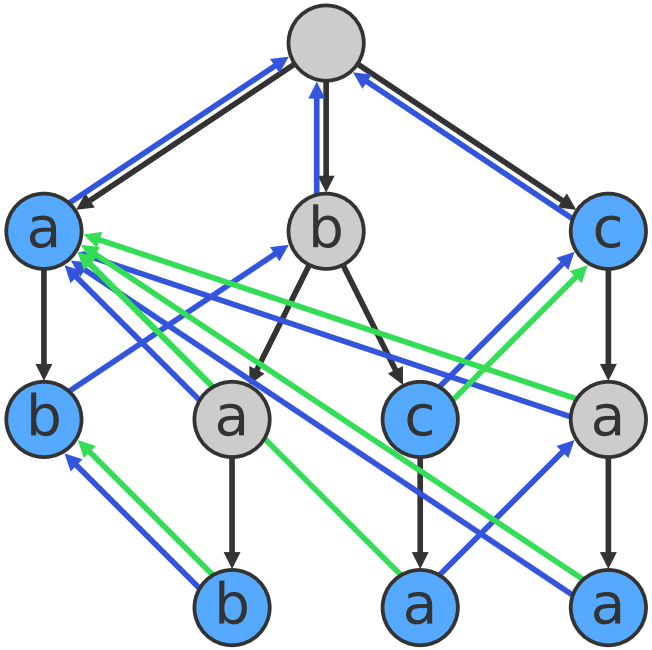
CC-BY Claudio Rochini, Wikimedia

- Raffinement des tries
- Fusion d'un nœud avec son nœud fils s'il est unique
- Espace mémoire considérablement réduit dans certaines applications
- Le branchement peut se faire bit par bit (quand on stocke des entiers), lettre par lettre, ou groupe de bits par groupe de bits (le nombre de bits par branchement est le radix)

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

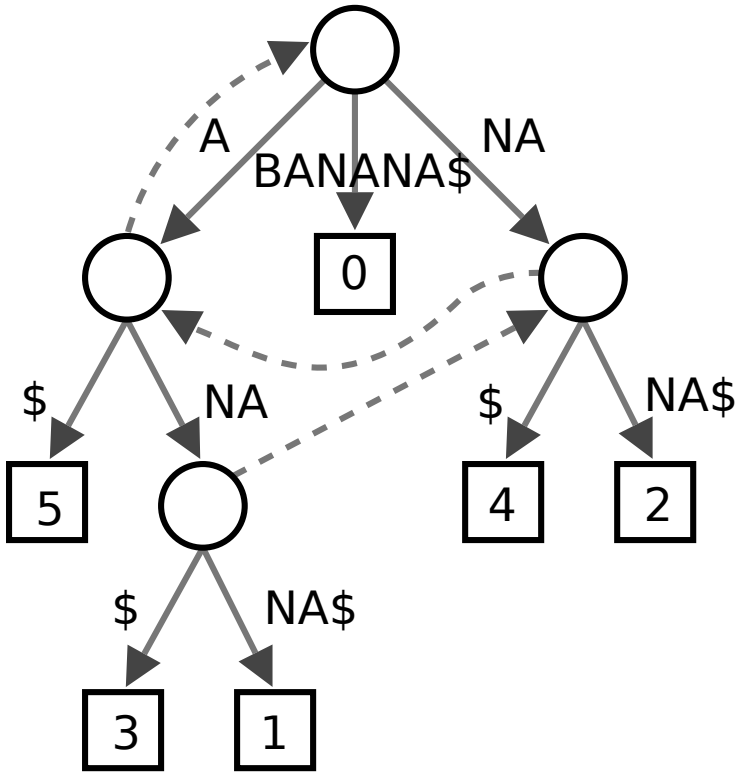
- Dictionnaire D de taille n contenant des mots de taille k, chaîne de taille l
- Applications répétées de Knuth–Morris–Pratt ? $O(n \times (k + l))$
- On peut faire mieux : généraliser l'algorithme de Knuth–Morris–Pratt à un trie arbitraire (et non une séquence de caractères) : $O(n \times k + l)$

Aho–Corasick



- On construit un trie du dictionnaire (en noir, les mots du dictionnaire sont sur fond bleu)
- On ajoute des pointeurs vers le plus grand suffixe strict présent dans le trie (en bleu)
- On ajoute la clôture transitive de ces pointeurs restreinte aux mots du dictionnaire
- Exemple : abccab donne successivement les mots a (parcours normal), ab (parcours normal), bc (après suivi du pointeur), c (clôture), c (après suivi du pointeur), a (clôture), ab (après suivi du pointeur)

Arbre des suffixes



- Trie Patricia des suffixes d'un mot donné (ici, BANANA, suivi d'un symbole spécial \$)
- Facilement constructible en temps $O(n^2)$ de droite à gauche
- Possible de le construire en temps $O(n)$ de gauche à droite, avec un mécanisme similaire aux pointeurs de Aho–Corasick
- Permet d'indexer une chaîne pour pouvoir ensuite efficacement 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 Java (`java.util.regex`) et dans la bibliothèque standard de C++ 2011 (`std::regex`)

Automates

- Une expression rationnelle peut être traduite en un automate fini nondéterministe en temps linéaire (algorithme de Thompson, résulte en des transitions spontanées) ou quadratique (algorithme de Glushkov, plus petit nombre d'états)
- Reconnaître si une chaîne de caractères 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 caractères de longueur n est acceptée par un automate déterministe à m états est en $O(n)$

Expressions rationnelles et expressions rationnelles

- Ce que Java ou C++ appellent expression rationnelle n'est pas une expression rationnelle :
  - Références arrières (permettant d'indiquer qu'une sous-chaîne se répète)
  - Opérateur $*$ glouton et $*?$ réticent
- Du coup, les implémentations des expressions rationnelles n'utilisent pas des automates, mais du backtracking
- Souvent beaucoup moins efficaces ! Exponentiel en la taille de l'expression rationnelle dans les cas pathologiques

Plan

En résumé

- Déterminer si une chaîne est dans un ensemble de chaînes : table de hachage
- Déterminer les chaînes d'un ensemble dont une chaîne est préfixe : trie, arbre radix
- Recherche d'une petite sous-chaîne d'une chaîne : implémentation native du langage de programmation
- Recherche d'une sous-chaîne de taille non négligeable d'une longue chaîne : Knuth–Morris–Pratt (ou Boyer–Moore, non traité ; KMP est meilleur quand l'alphabet est petit, p. ex., ACTG de l'ADN)
- Recherche d'un ensemble de sous-chaînes dans une chaîne : Aho–Corasick ou arbre des suffixes (Aho–Corasick indexe les sous-chaînes, l'arbre des suffixes indexe la chaîne)
- Recherche d'un motif complexe dans une chaîne : expression rationnelle ou parfois automate déterministe/nondéterminste compilé à partir du motif

Licence de droits d'usage

Contexte public } avec modifications

Par le téléchargement ou la consultation de ce document, l'utilisateur accepte la licence d'utilisation qui y est attachée, telle que détaillée dans les dispositions suivantes, et s'engage à la respecter intégralement.

La licence confère à l'utilisateur un droit d'usage sur le document consulté ou téléchargé, totalement ou en partie, dans les conditions définies ci-après et à l'exclusion expresse de toute utilisation commerciale.
Le droit d'usage défini par la licence autorise un usage à destination de tout public qui comprend :
– le droit de reproduire tout ou partie du document sur support informatique ou papier,
– le droit de diffuser tout ou partie du document au public sur support papier ou informatique, y compris par la mise à la disposition du public sur un réseau numérique,
– le droit de modifier la forme ou la présentation du document,
– le droit d'intégrer tout ou partie du document dans un document composite et de le diffuser dans ce nouveau document, à condition que :
  – L'auteur soit informé.

Les mentions relatives à la source du document et/ou à son auteur doivent être conservées dans leur intégralité.

Le droit d'usage défini par la licence est personnel et non exclusif.
Tout autre usage que ceux prévus par la licence est soumis à autorisation préalable et expresse de l'auteur : sitepedago@telecom-paristech.fr