

Cours n°10

Algorithmique du texte I

INTRODUCTION

Bibliographie

M. Crochemore, C. Hancart, T. Leroq, Algorithms on Strings, Cambridge University Press, 2007.

D. Gusfield, Algorithms on Strings, Trees, and Sequences, Cambridge University Press, 1997.

A. Apostolico, Z. Galil, Pattern Matching Algorithms, Oxford University Press Inc, 1997.

S.-S. Skiena, The Algorithm Design Manual (Hardcover), Springer, 2008.

Sites

www.cs.helsinki.fi/u/ukkonen/
www.cs.helsinki.fi/group/suds

Sommaire

1. Structures de données

- 1.1 Arbre littéral
- 1.2 Arbre des suffixes
- 1.3 Arbre généralisé des suffixes

2. Applications

- 2.1 Recherche exacte de mots
- 2.2 Recherche approchée de mots
- 2.3 Recherche de facteurs communs entre mots

3. Bibliothèque SuDS/cst

1 ARBRE DES SUFFIXES

Arbre littéral (ou trie)

Définition

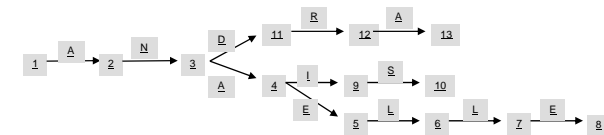
Arbre dont les arcs sont valués avec des symboles (e.g., caractères Unicode),
Symboles différents entre les fils d'un même sommet

Valuation d'un chemin

Mot correspondant à la concaténation des valuations des arcs du chemin

Mots d'un arbre littéral

Ensemble des valuations des chemins partant de la racine et arrivant à un sommet terminal



Arbre littéral de ANAELLE, ANAIS et ANDREA

Arbre littéral des suffixes

Soit A_w arbre littéral du mot w

Suffixes d'un mot w

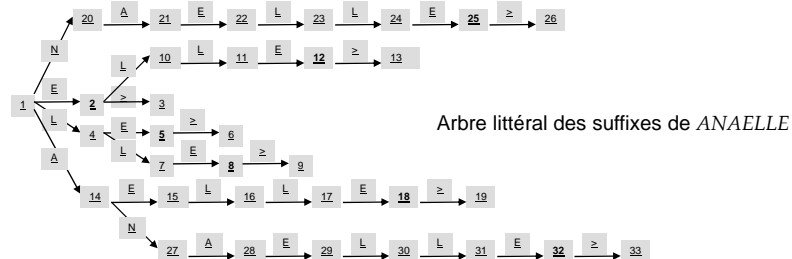
Ensemble des valuations des chemins de A_w arrivant à un sommet terminal

Préfixes d'un mot w

Ensemble des valuations des chemins de A_w partant de la racine

Arbre littéral des suffixes

Arbre littéral dont les mots sont les suffixes d'un mot w



Arbre des suffixes

Structure compressée de l'arbre littéral des suffixes

Soit un W un mot de taille n (e.g., les pages d'un site web)

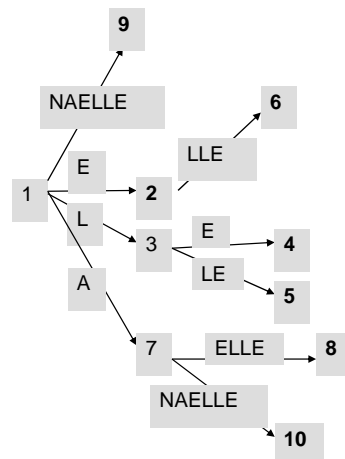
Cardinal(arbre littéral des suffixes de W) = n^2

Cardinal(arbre des suffixes de W) est en $O(n)$ ($\leq 20n$)

Historique

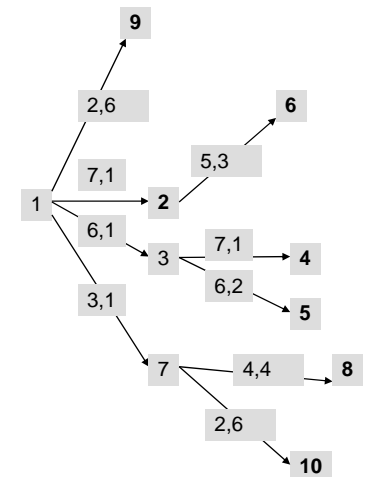
- | | |
|----------------|---|
| 1973 Wiener | Premier algorithme de construction en un temps linéaire et une mémoire quadratique (algorithme de l'année 1973) |
| 1976 McCreight | Algorithme off-line de construction en un temps linéaire et une mémoire linéaire |
| 1992 Ukkonen | Algorithme on-line (temps réel) de construction en un temps linéaire et une mémoire linéaire |
| 1999 Kurtz | Réduction de la mémoire nécessaire |
| 2007 Sadakane | Réduction de la mémoire nécessaire |

Arbre des suffixes du mot ANAELLE



Arbre des suffixes compacté du mot ANAELLE

Remplacement des sous-mots par deux entiers (début, taille)



1 ARBRE DES SUFFIXES

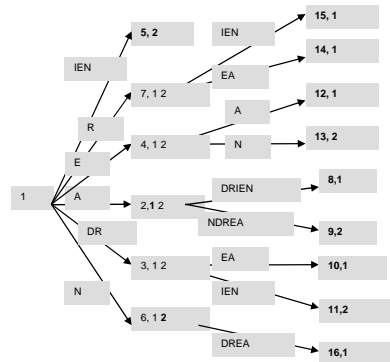
Arbre généralisé des suffixes

Arbre des suffixes pour un ensemble de mots (W1, W2, ..)

Variante de l'algorithme de construction,
Représentation compressée avec 3 entiers (débit, taille, numéro du mot)

Suffixes des mots ADRIEN et ANDREA

A (1)
ADRIEN(2)
ANDREA(1)
DREA(1)
DRIEN(2)
EA (1)
EN(2)
IEN(2)
N (2)
NDREA(1)
REA (1)
RIEN(2)



2 APPLICATIONS

Recherche exacte de mots - Complexité

Recherche de S dans W (longueur (S) = m, longueur (W) = n)

(1) Algorithme naïf

```

i := 0;
1 :   j := 0;
      tant que S[j] ≠ W[i+j] et j < m et (i+j) < n
      j := j + 1 ;
fin tant que
si j = m le motif est trouvé enregistrer la position trouvée (i)
sinon si i = n la recherche est terminée
      sinon incrémenter i et aller à l'étape 1 :

```

Complexité en $O(m * n)$

(2) Algorithm KMP (Knuth-Morris-Pratt) 1975

Complexité en $O(m + n)$

(3) Algorithme arbre de suffixes

Complexité en $O(m)$

Exemple sur la recherche google « Sorbonne »

 $m = 8$

n = 2 000 000 000 000 000 (2 millions de milliards de caractères)

2 APPLICATIONS

Recherche exacte de mots - Algorithme

Recherche de S dans W

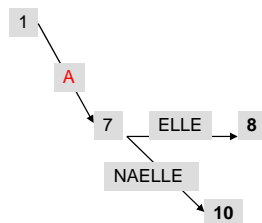
Soit A_W l'arbre des suffixes correspondant à W

Recherche d'un sommet x de A_W où S est une sous-chaîne de la valuation du chemin de la racine à x

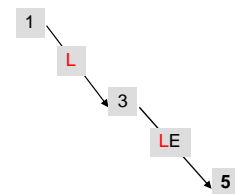
Si x n'existe pas alors il n'y a aucune occurrence de S dans W

Si x existe, il y a autant d'occurrences de S que de feuilles au sous-arbre x

Recherche des positions des occurrences de S en remontant des feuilles



2 Occurrences de A



1 Occurrence de LL

2 APPLICATIONS

Recherche approchée de mots

Recherche de S dans W avec k modifications au maximum

Temps de recherche linéaire par rapport à la taille du mot cherché S et au nombre de modifications (k)

Trois types de modifications (distance de Levenshtein)

Suppression d'un caractère	$c \rightarrow \epsilon$
Insertion d'un caractère	$\epsilon \rightarrow c$
Substitution d'un caractère	$c_1 \rightarrow c_2$

B O U L A N G E R
| | |
O R A N G E

Programmation dynamique entre S et l'arbre de suffixes

Recherche de facteurs communs entre mots

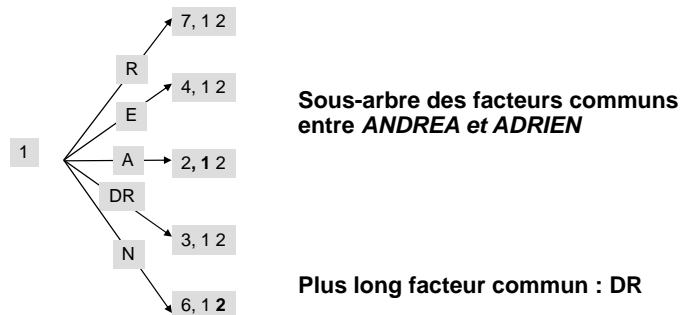
Recherche des facteurs communs entre les mots W1, W2, .

Suite de caractères présent dans plusieurs mots.

Construction de l'arbre généralisé des suffixes de ces mots

Extraction du sous-arbre comprenant les sommets étiquetés avec tous les mots

Parcours en profondeur du sous-arbre



Recherche des plagats

13

Master Langue et Informatique – Programmation générique et conception objet– Claude Montacié

Autres applications en temps linéaire

Recherche du plus long sous-mot de W apparaissant au moins q fois

Recherche des occurrences les plus fréquentes des sous-mots de W

Recherche de séquences cycliques de sous-mots de W

Parcours de l'arbre de suffixes,

Applications importantes en génomique, en musicologie, et en linguistique computationnelle

14

Master Langue et Informatique – Programmation générique et conception objet– Claude Montacié

Projet Succint Data Structure (SuDS)/cst

Implémentation d'un arbre de suffixe compressé

Arbre de suffixes d'un génome humain sur 8.8 GB,

Librairie en C++

Construction de l'arbre de suffixes

```
SSTree (uchar *W, ulong n, bool deletetext,
unsigned samplerate, io_action IOaction, const
char *filename)
```

W pointeur sur le texte

n taille du texte.

deletetext destruction du texte après construction (non par défaut)

samplerate Taux de compression pour les vecteurs de suffixes (non par défaut)

io_action lecture ou écriture sur fichier de l'arbre de suffixes (non par défaut)

filename Nom du fichier (rien par défaut)

15

Master Langue et Informatique – Programmation générique et conception objet– Claude Montacié

Bibliothèques de fonctions membres (1)

Parcours dans l'arbre de suffixes

ulong root ()

Nœud racine

bool isleaf (ulong v)

Est-ce que le nœud v est une feuille

ulong child (ulong v, uchar c)

Fils de v commençant par le caractère c

ulong firstChild (ulong v)

Premier fils de v

ulong sibling (ulong v)

Prochain frère de v

ulong parent (ulong v)

Parent de v

ulong depth (ulong v)

Profondeur du nœud v (en caractères)

ulong nodeDepth (ulong v)

Profondeur du nœud v (en nœuds)

...

Accès au texte

uchar edge (ulong v, ulong i)

ième caractère du nœud v

uchar* edge (ulong v)

chaîne de caractères du nœud v

uchar* pathlabel (ulong v)

chaîne de caractères de la racine au nœud v

16

Master Langue et Informatique – Programmation générique et conception objet– Claude Montacié

Bibliothèques de fonctions membres et de fonctions (2)

Affichage de l'arbre des suffixes

```
void PrintTree (ulong v, int d)
```

Affichage des chaînes de caractères associées aux noeuds à partir du noeud v sur une profondeur d

Recherche dans l'arbre des suffixes

```
ulong search (uchar *M, ulong m)
```

Noeud correspondant à la recherche du mot M de taille m dans l'arbre de suffixes (0 en cas d'échec)

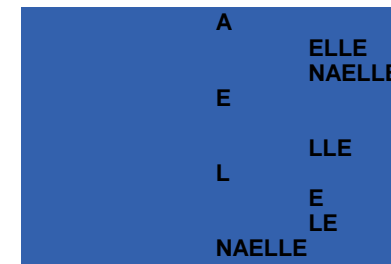
Recherche du plus long facteur commun entre deux mots A et B

```
string lcss (string A, string B)
```

Construction et affichage de l'arbre des suffixes

```
cout << "lecture au clavier du mot" << endl;
string A; cin >> A;
SSTree *sst = new SSTree((uchar*)(A.c_str()), A.size()+1);
```

```
// Affichage de l'arbre de suffixes
sst->PrintTree(sst->root(), 0);
```



Arbre des suffixes de ANAELLE

Recherche d'un mot dans l'arbre de suffixes

```
int main() {
    string A; char c;
    ifstream fs("livres/adolphe.txt");
    while (!fs.eof()) { fs.get(c); A += c;}
    fs.close();
    SSTree *sst = new SSTree((uchar*)(A.c_str()), A.size()+1);
    cout << "lecture au clavier du mot" << endl;
    string B; cin >> B;
    ulong i = sst->search((uchar*)(B.c_str()), B.size());
    cout << "le numéro du noeud est " << i << endl;
    cout << "il correspond au texte " << sst->pathlabel(i) << endl;
    ulong d = sst->textpos(i);
    cout << "dont la position est " << d << endl;
}
```

```
lecture au clavier du mot
Adolphe
le numéro du noeud est 103340
il correspond au texte Adolphe
dont la position est 30590
```

Calcul du plus long facteur commun

```
#include <iostream>
#include <fstream>
#include <string>
#include "SuDS/SSTree.h"
#include "SuDS/Tools.h"

using namespace std;

int main() {
    string A, B; char c;
    ifstream fs1("livres/candidate.txt");
    while (!fs1.eof()) { fs1.get(c); A += c;}
    fs1.close();
    ifstream fs2("livres/l'histoire de l'archiduc albert.txt");
    while (!fs2.eof()) { fs2.get(c); B += c;}
    fs2.close();
```

```
cout << "le plus long facteur commun est " << lcss(A,B) << "\n\n";
}
```

```
le plus long facteur commun entre les deux livres est
« ils se mirent tous à genoux »
```

Mémoire utilisée : 12 Mo