—) C7 - Algorithmique du texte : exemples et applications (démo soutenance)

mettre des "(Admis)"

Notations Σ alphabet = ens. fini

$w \in \Sigma^*$ $w[i]$ i-ème lettre
$w[i..j]$ facteur de i à j
w_i préfixe de longueur i

Motivations :
- recherche et remplacement : barkour de bit.
- Traitement automatique de journaux / scripts.
- détection information : audio, bio-informatique
- transmission de l'information
- compilation

I - RECHERCHE DE MOTIFS

1 - Motif exact

Texte $T \in \Sigma^*$ $|T|=n$ Motif $P \in T^*$ $|P|=m \leq n$

a - Algorithme naïf

```
k ← 0
Tant que k+m ≤ n-1
| Si P = T[k+m..k+m]
| Alors OCCURENCE(k)
| k++
Fin Tantque
```

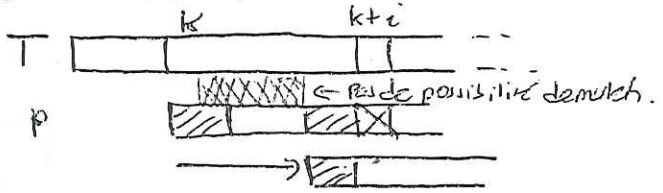
Complexité pire cas (meilleur) $O(mn)$ en temps.

b - Utilisation des bords : algorithme de Morris-Pratt.

Déf Bord (u) = plus long préfixe strict de u aussi suffixe de u.

Exemple Bord(ababa) = aba (aba / aba)

Idée : être plus astucieux en cas d'échec en utilisant la structure de P.



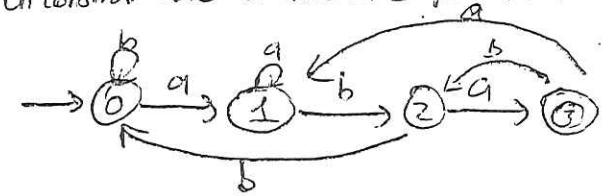
↳ Nécessité de calculer la table des bords de P en temps $\Theta(m|\Sigma|)$.

Remarque : cette méthode est encore améliorée par l'algorithme de Knuth-Morris-Pratt.

c - Automate des occurrences

Idée : Trouver P dans T c'est trouver tous les T_i ayant P pour suffixe ie. $\{i / T_i \in \Sigma^* P\}$.

On construit donc un automate par exemple P = aba



On note $\phi(T_i)$ l'état après lecture de $T[0..i] = T_i$ et $\sigma(T_i)$ le plus long préfixe de P suffixe de T_i.

Théorème | $\forall i \in \{0,1,\dots,n\}$ $\phi(T_i) = \sigma(T_i)$
L'automate des occurrences est minimal pour reconnaître $\Sigma^* P$.

DEV 1

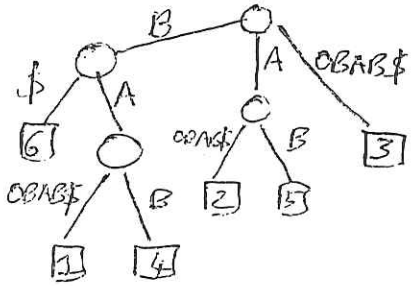
Extension plusieurs mots = Aho et Corasick squelette.

d - Arbre des suffixes

Idée : texte prédéfini dans lequel on va faire un grand nombre de recherches inconnues à l'avance : grand prétraitement.

Définition L'arbre des suffixes du texte T est un arbre enraciné à n feuilles (par chaque préfixe) étiquetté par des facteurs de T tels que la concaténation des étiquettes de la racine à la feuille k est le suffixe d'indice k.

Exemple T = BAOBAB$
1 2 3 4 5 6



Théorème [Ukkonen 95] L'arbre des suffixes peut être construit en temps linéaire $O(n)$!!!

La recherche de P (exacte) s'effectue alors en temps $O(m)$ et toutes les occurrences de P en parcourant le sous-arbre.

e) Bilan

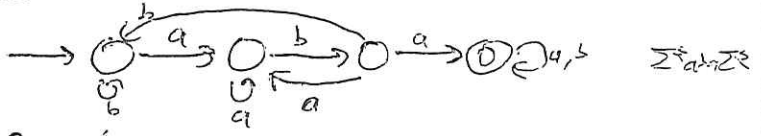
	Prétraitement	Exécution	Espace
Naïf	—	$O(mn)$	$O(1)$
MP	$O(m\|\Sigma\|)$	$O(n)$	$O(m)$
KMP	$O(m)$	$O(n)$	$O(m)$
Automate	$O(m\|\Sigma\|)$	$O(n)$	$O(m\|\Sigma\|)$
Arbre	$O(n)$	$O(m)$	$O(n \log n)$

2) Probabilité d'occurrence d'un motif

Idée Source d'élaboire (α : A 25%, B 17%, ...) et T_n issu de cette source. On décide BONJOUR, est-ce significatif ?

On peut modifier l'automate des occurrences et le transformer en chaîne de Markov

Ex : aba a 1/4 b 3/4



n = 0	1	0	0	0
n = 1	3/4	1/4	0	0
⋮				

Rq cela fonctionne car l'automate est déterministe.

3) Cas général : expressions régulières

En informatique, on cherche souvent P sous forme d'expression régulière.

Idée : construire l'automate et le déterminiser/minimiser

Bm : coût éventuellement exponentiel.

logiciel : GREP

II COMPARAISONS

→ Très utile en biologie : homologies → orthographe
→ traitement du signal → détection de virus.

1) Distances utilisées

- Mots de même taille = distance de Hamming (nbr de différences)
- Distance d'édition ou de Levenshtein : nbr minimum de substitutions, insertions, suppressions.
- Matrices de substitutions : Blosum en bio info.

DEV2

2) Calcul de la distance d'édition par programmation dynamique.

$T[i,j] = Lev(x[0..i], y[0..j])$

$T[-1,-1] = 0$

$T[i,-1] = T[i-1,-1] + 1$

$T[-1,j-1] = T[-1,j-1] + 1$

$$T[i,j] = \min \begin{cases} T[i-1,j-1] + 1 & \text{(substitution)} \\ T[i-1,j] + 1 & \text{(suppression)} \\ T[i,j-1] + 1 & \text{(insertion)} \end{cases}$$

Utilisation : correction orthographe

3) Détection de répétitions

Ex : on cherche les facteurs qui se répètent 2x dans BANANAS

① construire l'arbre des suffixes

② parcourir l'arbre et compter le nombre de feuilles de chaque sous arbre (récursivement)

③ Parcours en profondeur, seulement les nœuds ≥ 2

ex : B et BA et A (et E).

III Compression : codage de Huffman

On a soit un texte écrit avec Σ ou alors une source aléatoire ~~émettant~~ émettant des lettres de Σ. On note f_i la fréquence / probabilité d'occurrence de la lettre i.

On veut encoder en binaire Σ de façon à minimiser le nombre de bits : $\sum_{a \in \Sigma} f_a \times$ longueur a.

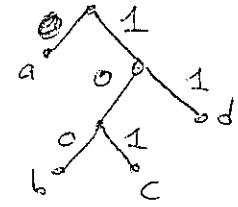
Huffman : on construit un arbre d'encodage de manière gloutonne.

① $|\Sigma|$ feuilles de poids f_i.

Répéter ② Trouver les deux sous arbres de poids minimal et fusionner, sommer les coût des sous-arbres.

Ex

Σ	f_i
a	20
b	20
c	3
d	37



Théorie de l'information.

- Information d'une lettre i : $-\log_2 p_i$ (en bits)
- Entropie de la source : $H = -\sum_{x \in \Sigma} p_x \log_2 p_x$
  = information moyenne.
- Théorème de Shanon : on ne peut pas faire mieux que H.
- Huffman : codage optimal au sens de la théorie de l'information

Automate des occurences [CORMEN pp 915 - 9

Notations

$P \in \Sigma^*$ $T \in \Sigma^*$

$|P| = m \leq |T| = n$

$\exists$ "est suffixe de"

P_i : $i$ème préfixe

$\sigma(x) = \max\{n \mid P_n \exists x\}$

Automate $\mathcal{A}$

$Q = [\![0, m]\!]$

$I = 0$ $\quad F = \{m\}$

$\delta(q, a) = \sigma(P_q\, a)$

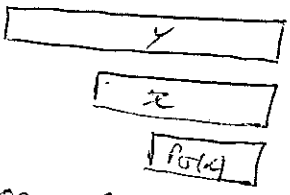
ϕ : $\begin{cases} \phi(\varepsilon) = 0 \\ \phi(xa) = \delta(\phi(x), a) \end{cases}$

Théorème $\forall i \in [\![0, n]\!]$, $\phi(T_i) = \sigma(T_i)$. En particulier $\mathcal{L}(\mathcal{A}) = \Sigma^* P$.

De plus $\mathcal{A}$ est minimal

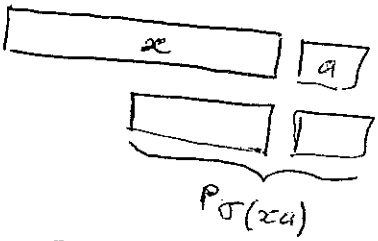
Démonstration Correction

Lemme 1 $\forall x \in \Sigma^* \; \forall y \in \Sigma^* \quad x \exists y \Rightarrow \sigma(x) \leq \sigma(y)$



$P_{\sigma(x)} \exists x$ et $x \exists y$ donc $P_{\sigma(x)} \exists y$ donc $\sigma(x) \leq \sigma(y)$

Lemme 2 $\forall x \in \Sigma^* \; \forall a \in \Sigma, \quad \sigma(xa) \leq \sigma(x) + 1$



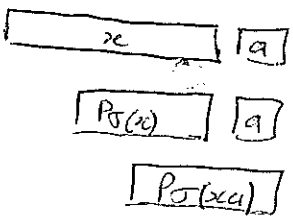
$P_{\sigma(xa)} \exists xa$

donc $P_{\sigma(xa)-1} \exists x$

donc $\sigma(xa) - 1 \leq \sigma(x)$

Lemme 3 $\forall x \in \Sigma^*, \; \forall a \in \Sigma, \quad \sigma(P_{\sigma(x)}\, a) = \sigma(xa)$

- $P_{\sigma(x)} \exists x \Rightarrow P_{\sigma(x)}\, a \exists xa \underset{\text{(lemme 1)}}{\Rightarrow} \sigma(P_{\sigma(x)}\, a) \leq \sigma(xa)$

- 

On a $P_{\sigma(xa)} \exists xa$

et $P_{\sigma(x)}\, a \exists xa$

donc $|P_{\sigma(xa)}| = \sigma(xa) \underset{\text{(lemme 2)}}{\leq} \sigma(x) + 1 \leq |P_{\sigma(x)}| + 1 = |P_{\sigma(x)}\, a|$

d'où $|P_{\sigma(xa)}| \leq |P_{\sigma(x)}\, a|$ et $P_{\sigma(xa)} \exists P_{\sigma(x)}\, a$

d'où $\sigma(xa) \leq \sigma(P_{\sigma(x)}\, a)$

ve du théorème Récurrence sur i

$i=0$ $T_0 = \varepsilon$ $\phi(\varepsilon)=0$ et $\sigma(\varepsilon) = \max\{n \mid P_n \rfloor \varepsilon\} = 0$ $\phi(T_0) = \sigma(T_0)$

$i \Rightarrow i+1$ $\phi(T_{i+1}) = \phi(T_i a) = \delta(\phi(T_i), a) = \delta(\sigma(T_i), a) = \sigma(P_{\sigma(T_i)} a) \underset{\text{lemme 3}}{=} \sigma(T_i a)$

$a = T[i+1]$

$\sigma(T_i a) = \sigma(T_{i+1})$.

Minimalité

Soient i j qcq vérifiant $0 \leq i < j \leq m$

On pose $Q_k \in \Sigma^*$ définir par $P_k Q_k = P$

On note L_k le langage reconnu à partir de l'état k.

$\phi(P_j \cdot Q_j) = \phi(P) = m$ donc $Q_j \in L_j$

$\phi(P_i Q_j) < m$ car $|P_i Q_j| < m$ donc $Q_j \notin L_j$

Ainsi $i \not\equiv j$ dans l'équivalence de Nerode. Tous les états sont distinguables 2 à 2 donc $\mathcal{A}$ est minimal !