Leçon 907 Algorithmique du texte

I. Définitions

1. **Déf** Un **mot** est une suite finie d'éléments d'un **alphabet** fini A. L'ensemble des mots est noté A^*.

2. **Déf** Soit $w = w_1 \dots w_n \in A^*$. On appelle :

préfixe	$w_1 \dots w_k$	$1 \le k \le n$
suffixe	$w_k \dots w_n$	———
sous-mot	$w_{i_1} \dots w_{i_k}$	$1 \le i_1 < \dots < i_k \le n$
facteur	$w_i \dots w_j$	$1 \le i \le j \le n$

II. Recherche de motif

3. **Problème** de recherche de motif
Pour $X \subseteq A^*$ (motif) et $w \in A^*$ (texte) on cherche les occurrences de mots de X comme facteurs de w.

4. **Problème** de recherche de mot :
cas particulier $X = \{m\}$.

II.1 Recherche simple d'un mot

5. **Algorithme** naïf : pour chaque $i = 1, \dots, |w|-|m|+1$, on vérifie si $w[i \dots i+|m|-1] = m$

6. **Prop** La recherche naïve fonctionne en temps
$O(|m| \cdot (|w|-|m|))$ pire cas,
$\Theta(|w|-|m|)$ en moyenne.

7. **Algorithme** de Rabin-Karp :
on se munit d'une fonction de hachage H telle que $H(w[i+1 \dots j+1])$ se calcule en temps $O(1)$ à partir de $H(w[i \dots j])$.
À chaque itération, on teste si $H(w[i \dots i+|m|-1]) = H(m)$. La complexité en pire cas reste la même.

II.2. Recherche par automates

8. **Prop** La lecture d'un mot $w \in A^*$ par un automate fini déterministe représenté par matrice de transitions prend un temps $O(|w|)$.

9. **Structure** : Automate d'Aho-Corasick
Soit $X \subseteq A^*$ fini, on définit l'AFDC :
États : préfixes (des mots) de X
Initial : ε
Finaux : (préfixes de X) $\cap A^*X$
Transitions : $u \xrightarrow{a}$ plus grand suffixe de ua qui soit préfixe de X.

10. **Théorème** Cet automate reconnaît A^*X.

DFA

11. Exemple cf. Figure 1.
Remarquons que cet AFDC n'est pas minimal.

12. Algorithme : on peut construire cet automate en temps $O(|X|)$ où $|X| = |w_1| + \dots + |w_n|$ $X = \{w_1, \dots, w_n\}$.

13. Application à la recherche d'un motif X (X fini) dans $w \in A^*$: on lit w avec l'automate et on regarde quand un état final est traversé.
Temps $O(|w| + |X| + \text{nombre d'occurrences})$

14. Structure : Automate de Simon cas particulier $X = \{m\}$ de Aho-Corasick.

15. Théorème L'automate de Simon est l'AFDC minimal pour A^*m.

16. Algorithme de Knuth-Morris-Pratt : optimisation de la méthode présentée précédemment en calculant implicitement l'automate de Simon.
Complexité temporelle : $O(|m| + |w|)$.

III. Structures de données pour le texte

17. Structure d'arbre dictionnaire (trie) pour un ensemble fini de mots $X \subseteq A^*$: les nœuds sont des préfixes de X et ua est un fils de u par une branche étiquetée a pour tous $(u, a) \in A^* \times A$ tels que ua préfixe de X.

18. Remarque : cet arbre peut être vu comme un automate reconnaissant X, sous-graphe de l'automate d'Aho-Corasick.

19. Exemple cf. Figure 2, à comparer avec la figure 1.

20. Application Structures d'ensemble et de tableau associatif clé-valeur dont les clés sont des mots.

21. Structure Arbre des suffixes
IP s'agit de l'arbre dictionnaire pour l'ensemble des suffixes d'un mot.

22. Algorithme d'Ukkonen
Une représentation compacte de l'arbre des suffixes de $w \in A^*$ est calculable en temps $O(|w|)$.

24. Algorithme de tri des suffixes
La permutation $\sigma \in \mathfrak{S}_n$ telle que
$w[\sigma(1)\ldots n] < \ldots < w[\sigma(n)\ldots n]$
pour l'ordre lexicographique ($w \in A^*$, $|w|=n$)
peut être calculée en temps $O(n \log n)$

DEVN

25. Structure de table des suffixes d'un mot
À $\sigma \in \mathfrak{S}_n$ définie ci-dessus, on ajoute un tableau mémorisant la longueur du plus grand préfixe commun de $w[\sigma(i)\ldots n]$ et $w[\sigma(i+1)\ldots n]$, qu'on calcule en temps linéaire.

26. Application Arbre des suffixes et table des suffixes s'appliquent généralement aux mêmes problèmes ; par exemple, l'un ou l'autre permettent de trouver le plus long facteur palindronique d'un mot.

IV. Alignements

27. Déf L'insertion d'une lettre a en i-ième position dans le mot w, $|w| \geq i$, donne $w[1..i-1] \cdot a \cdot w[i..|w|]$.
On définit de même suppressions et modifications.

28. Déf La distance d'édition $d(w,w')$ est le nombre minimum d'insertions, suppressions et modifications nécessaires pour passer de w à w'.
On peut également considérer la distance d' sans ~~suppressions~~ modifications.

29. Prop $\forall w, w' \in A^*$, $d'(w,w') = |w| + |w'| - 2|$plus long sous-mot commun$|$.
La notion analogue pour d est celle d'alignement
Cf. figures 3 et 4.

30. Algorithme $d(w,w')$ et $d'(w,w')$ peuvent être calculés en ~~temps~~ $O(|w|.|w'|)$ et espace $O(\min(|w|,|w'|))$ par programmation dynamique.

31. Algorithme de Hirschberg
Calcul de l'alignement optimal ou du plus long sous-mot commun avec la même complexité asymptotique.

Figure 1 Automate d'Aho-Corasick pour {aa, ba}

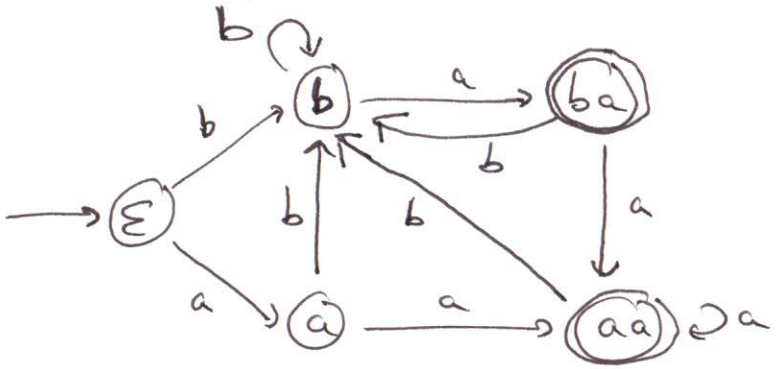

Figure 2 Arbre dictionnaire pour {aa, ba}

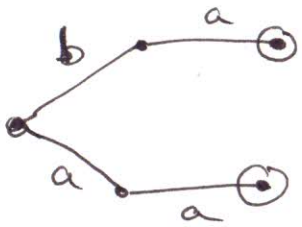

Figure 3 Plus long sous-mot commun

ALGEBRE
ALGORITHME

Figure 4 Alignement optimal

ALG E BRE
ALGORITHME
↳ modification