

Recherche d'un ensemble fini de mots

Thierry Lecroq
Université de Rouen

Le problème

Localiser toutes les occurrences d'un ensemble fini de k mots $X = \{ x_0, x_1, \dots, x_{k-1} \}$ dans un texte y de longueur n .

Soit $|X| = |x_0| + |x_1| + \dots + |x_{k-1}|$.

Construction de $T(X)$

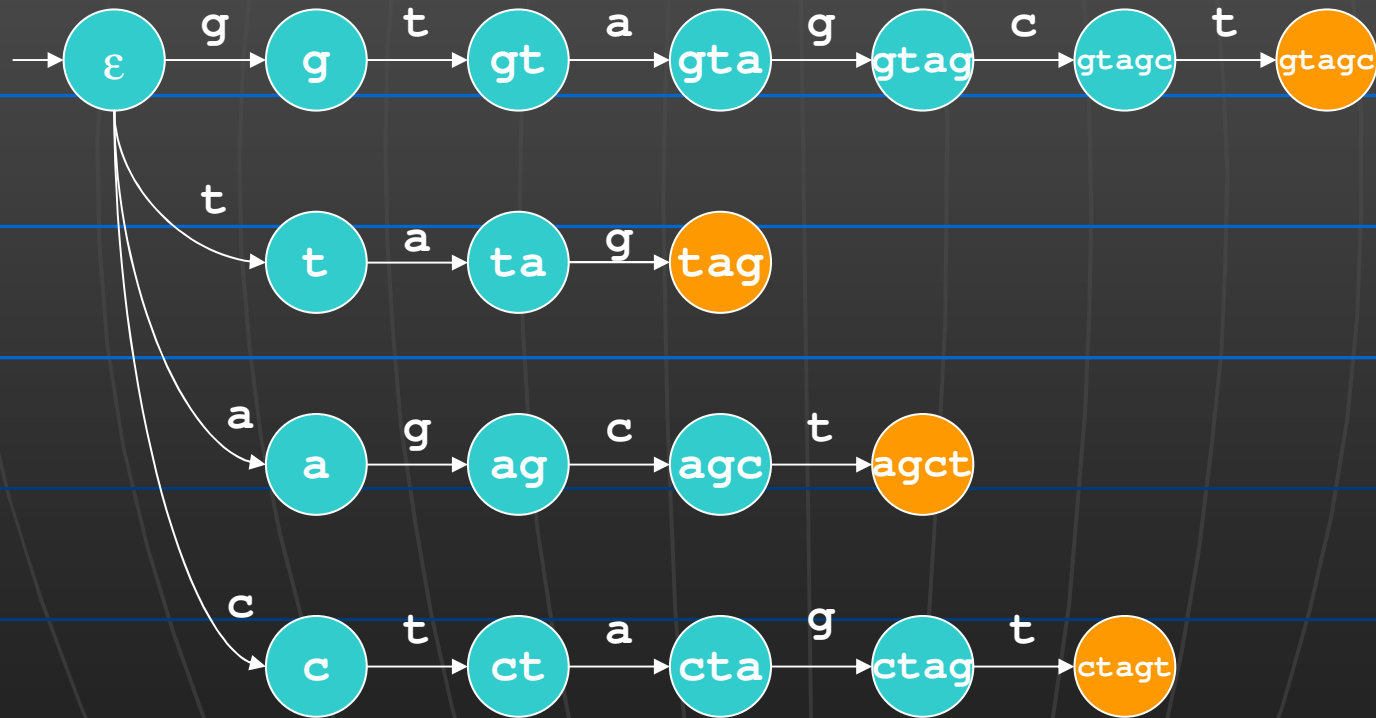
On commence par construire l'arbre (*trie*)

$T(X) = (Q, q_0, F, \delta)$ tel que :

- $Q = \text{Préf}(X)$
- $q_0 = \varepsilon$
- $F = X$
- pour $u \in \text{Préf}(X)$ et $a \in A$
 $\delta(u, a) = ua$ si $ua \in \text{Préf}(X)$ et
 $\delta(u, a)$ est indéfini sinon.

Exemple

$X = \{ \text{gtagct}, \text{tag}, \text{agct}, \text{ctagt} \}$

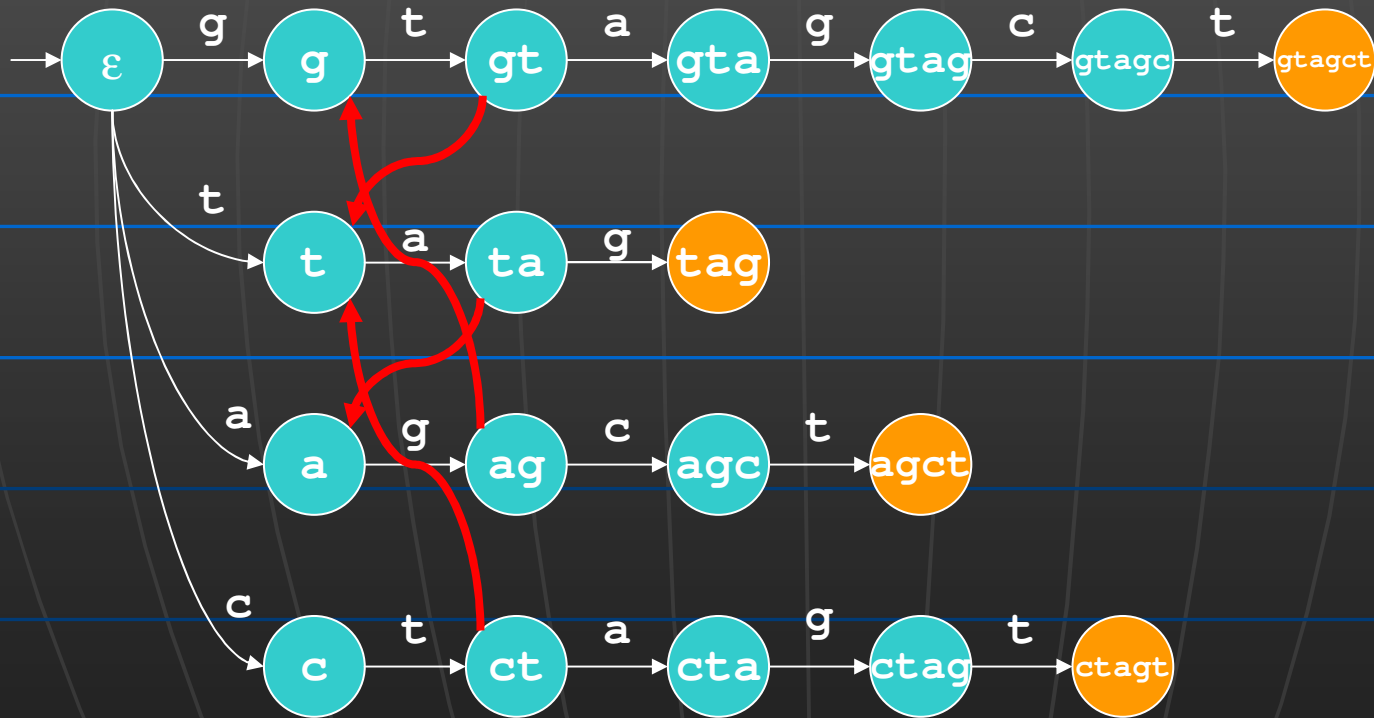


Suite de la construction

- On associe une fonction de sortie à chaque état terminal : $sortie(x) = \{ x \}$ si $x \in X$
- On crée une boucle sur l'état initial : $\delta(\varepsilon, a) = \varepsilon$ pour $a \in A$ et $a \notin Préf(X)$
- On associe à chaque état un état suppléant : $sup(q) = u$ où u est le plus long suffixe propre de q qui appartient à $Préf(X)$
- On complète la fonction de sortie : si $sup(q) = u$ alors $sortie(q) = sortie(q) \cup sortie(u)$

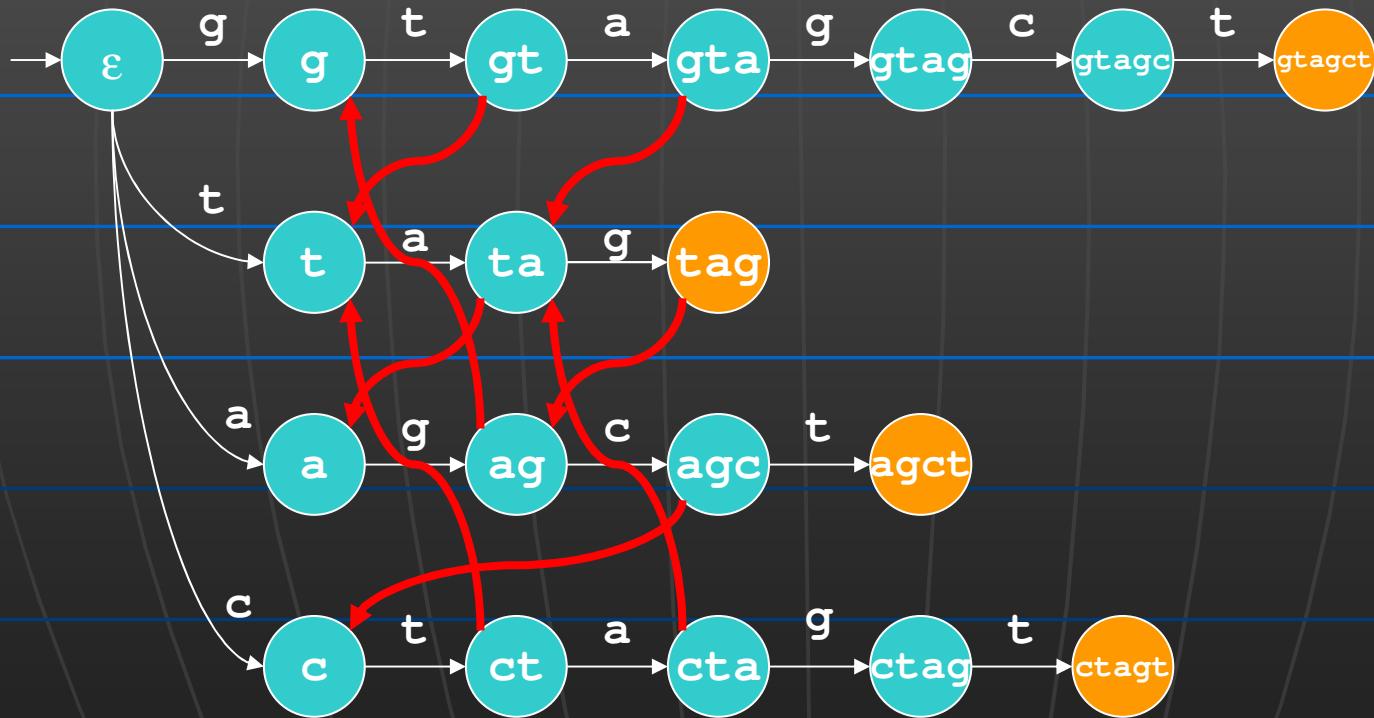
Exemple

$X = \{ \text{gtagct}, \text{tag}, \text{agct}, \text{ctagt} \}$



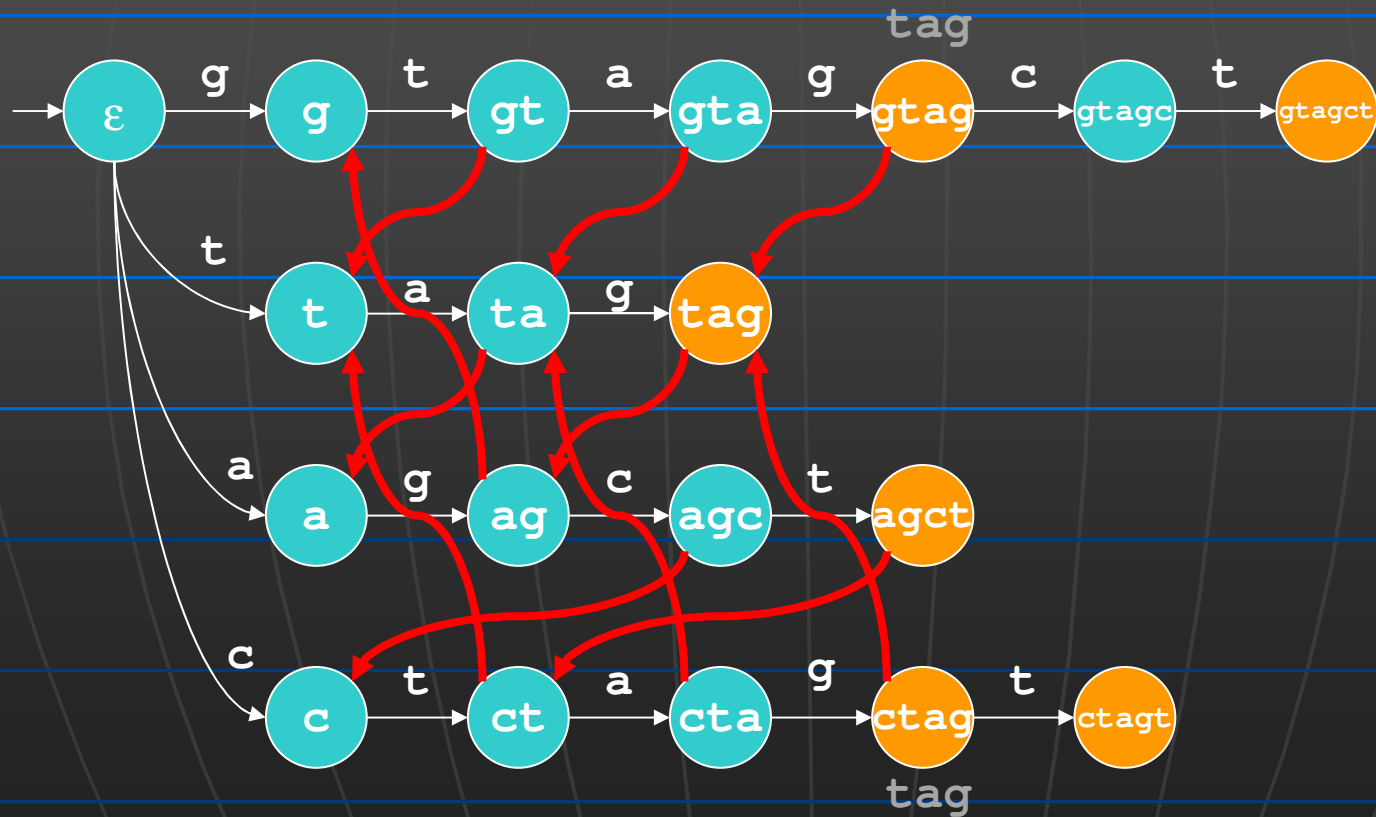
Exemple

$X = \{ \text{gtagct}, \text{tag}, \text{agct}, \text{ctagt} \}$



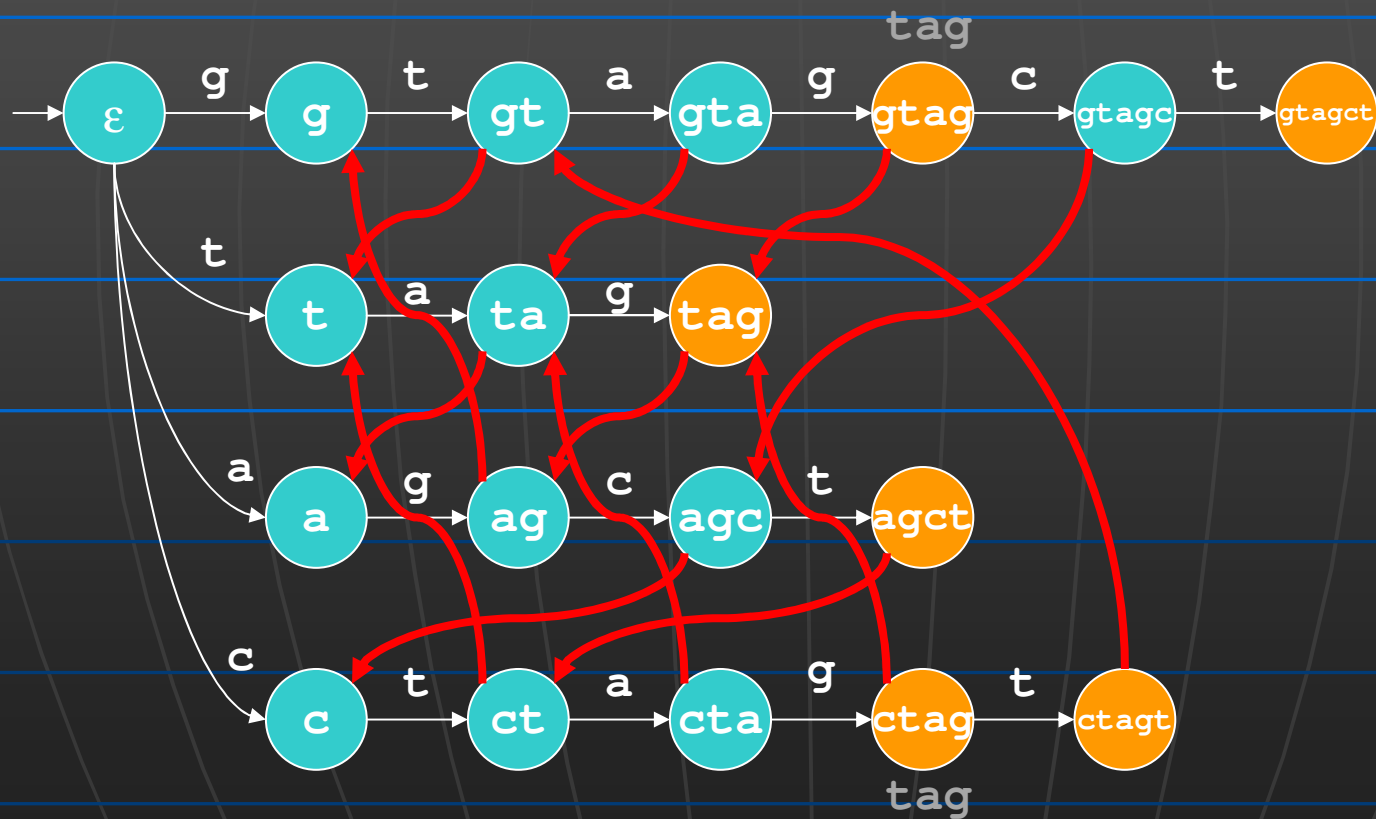
Exemple

$X = \{ \text{gtagct}, \text{tag}, \text{agct}, \text{ctagt} \}$



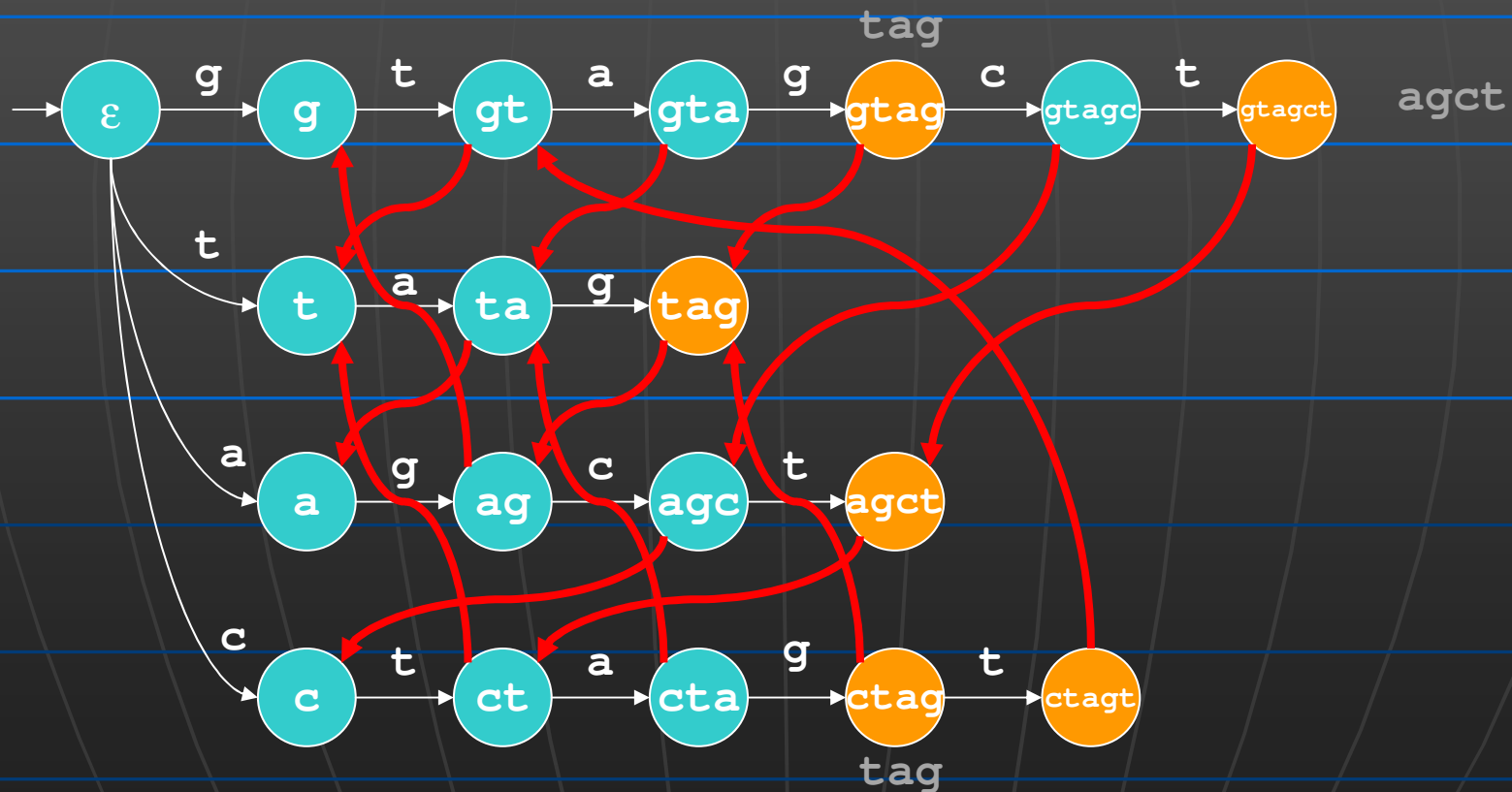
Exemple

$X = \{ \text{gtagct}, \text{tag}, \text{agct}, \text{ctagt} \}$



Exemple

$X = \{ \text{gtagct}, \text{tag}, \text{agct}, \text{ctagt} \}$



La construction de la fonction de suppléance est effectuée par un parcours en largeur de l'arbre $T(X)$ donc en utilisant une file.

algo PRE-AC(X, k)

créer l'état q_0

pour $a \in A$ **faire**

$\delta(q_0, a) \leftarrow a$

pour $i \leftarrow 0$ à $k-1$ **faire**

ENTRER($X[i], q_0$)

COMPLETER(q_0)

retourner q_0

algo ENTRER(x, e)

$i \leftarrow 0$

tantque $i < |x|$ et $\delta(e, x[i])$ est définie **faire**

$e \leftarrow \delta(e, x[i])$

$i \leftarrow i + 1$

tantque $i < |x|$ **faire**

créer un état s

$\delta(e, x[i]) \leftarrow s$

$e \leftarrow s$

$i \leftarrow i + 1$

$sortie(e) \leftarrow \{ x \}$

algo COMPLETER(e)

$f \leftarrow$ file vide

$\ell \leftarrow$ liste des transitions (e, a, p) telles que $p \neq q_0$

tantque ℓ est non vide **faire**

$(r, a, p) \leftarrow$ PREMIER(ℓ)

$\ell \leftarrow$ SUIVANT(ℓ)

ENFILER(f, p)

$sup(p) \leftarrow q_0$

tantque f est non vide **faire**

$r \leftarrow$ DEFILER(f)

$\ell \leftarrow$ liste des transitions (r, a, p)

tantque ℓ est non vide **faire**

$(r, a, p) \leftarrow$ PREMIER(ℓ)

$\ell \leftarrow$ SUIVANT(ℓ)

ENFILER(f, p)

$s \leftarrow sup(r)$

tantque $\delta(s, a)$ est non définie **faire**

$s \leftarrow sup(s)$

$sup(p) \leftarrow \delta(s, a)$

$sortie(p) \leftarrow sortie(p) \cup sortie(sup(p))$

Complexité de la phase de prétraitement

La phase de prétraitement s'effectue en temps $O(|X|)$.

algo AC(X, k, y, n)

$e \leftarrow \text{PRE-AC}(X, k)$

pour $j \leftarrow 0$ à $n-1$ **faire**

tantque $\delta(e, y[j])$ est non définie **faire**

$e \leftarrow \text{sup}(e)$

$e \leftarrow \delta(e, y[j])$

si *sortie*(e) **alors**

reporter une occurrence des éléments
de *sortie*(e) en position droite j

Complexité

L'algorithme de Aho-Corasick trouve toutes les occurrences d'un ensemble X de k mots dans un texte y de longueur n en $O(|X| + n)$.

Référence

A.V. Aho et M.J. Corasick

Efficient string matching: an aid to bibliographic search

Communications of the ACM 18(6) (1975) 333-340