Recherche d'un ensemble fini de mots

Thierry Lecroq

Université de Rouen
FRANCE

Le problème

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

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

Une première solution

Appliquer k fois un algorithme de recherche exact d'un seul mot

Complexité : $O(kn)$.

Une deuxième solution

En $O(n)$ sur un alphabet constant.

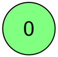
Construction de $T(X)$

On commence par construire l'arbre (*trie*) $T(X) = (Q, q_0, F, \delta)$

- $Q =$ *Préf*(X)
- $q_0 = \varepsilon$
- $F = X$
- pour $u \in$ *Préf*(X) et $a \in A$

$$\delta(u,a) = \begin{cases} ua & \text{si } ua \in \textit{Préf}(X) \\ \text{indéfini} & \text{sinon} \end{cases}$$

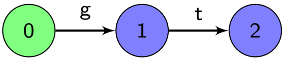
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



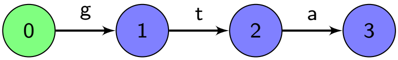
Exemple $X = \{\texttt{gtagct}, \texttt{tag}, \texttt{gagct}, \texttt{ctagt}\}$

$$0 \xrightarrow{g} 1$$

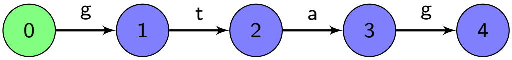
Exemple $X = \{\texttt{gtagct}, \texttt{tag}, \texttt{gagct}, \texttt{ctagt}\}$



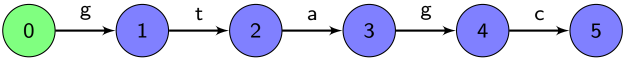
Exemple $X = \{\texttt{gtagct}, \texttt{tag}, \texttt{gagct}, \texttt{ctagt}\}$



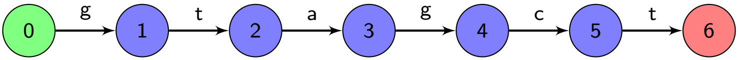
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



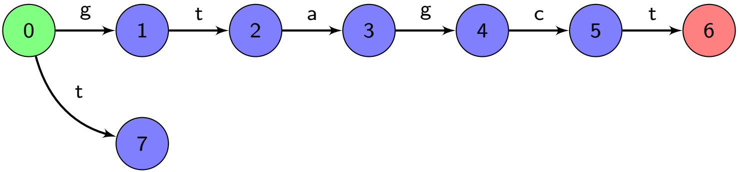
Exemple $X = \{\texttt{gtagct}, \texttt{tag}, \texttt{gagct}, \texttt{ctagt}\}$



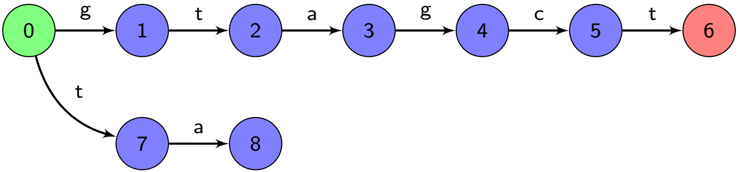
Exemple $X = \{\texttt{gtagct}, \texttt{tag}, \texttt{gagct}, \texttt{ctagt}\}$



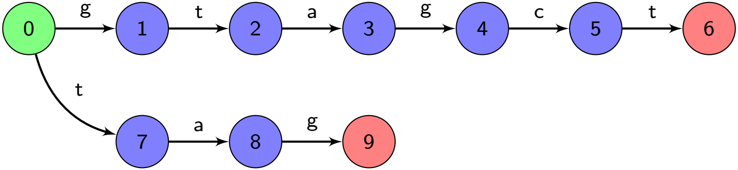
Exemple $X = \{\texttt{gtagct}, \texttt{tag}, \texttt{gagct}, \texttt{ctagt}\}$



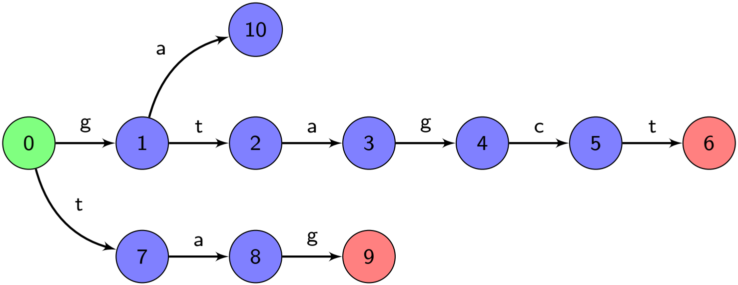
Exemple $X = \{\texttt{gtagct}, \texttt{tag}, \texttt{gagct}, \texttt{ctagt}\}$



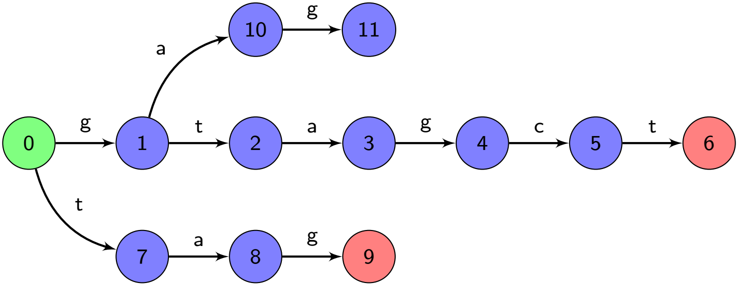
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



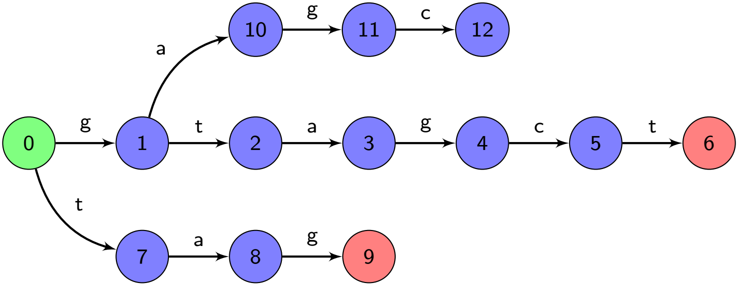
Exemple $X = \{\texttt{gtagct}, \texttt{tag}, \texttt{gagct}, \texttt{ctagt}\}$



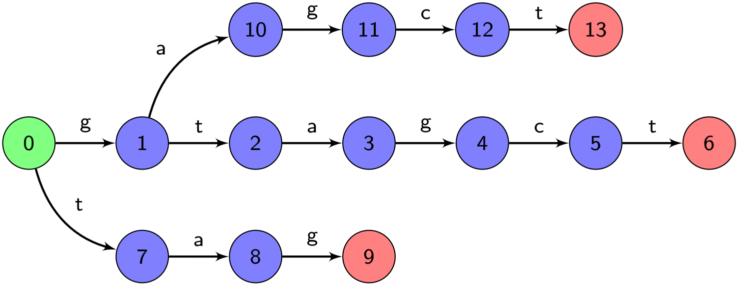
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



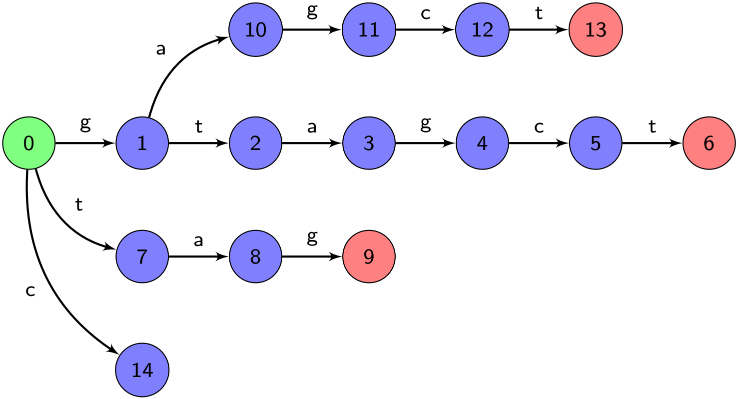
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



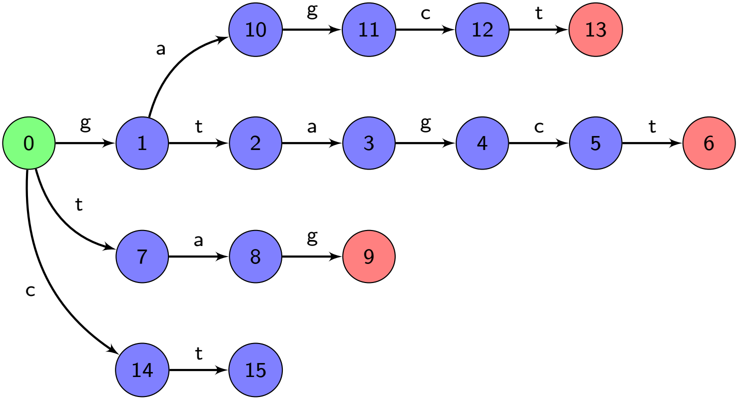
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



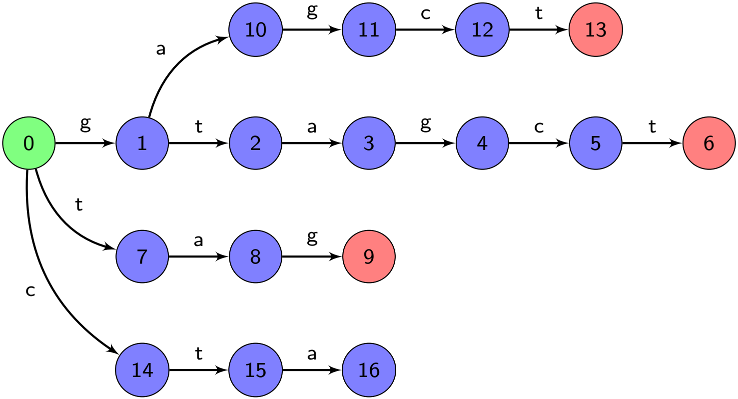
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



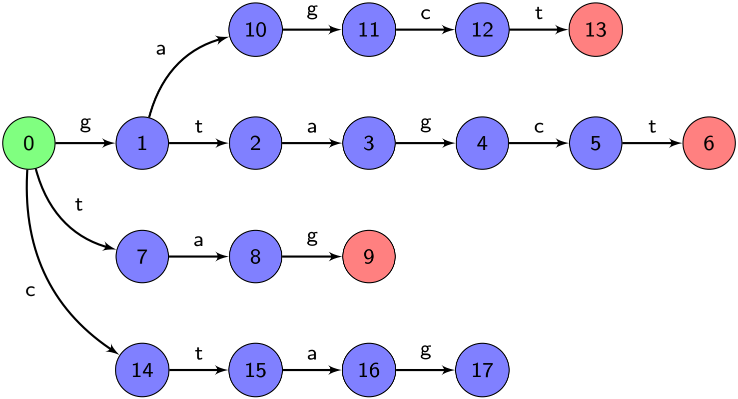
Exemple $X = \{\texttt{gtagct}, \texttt{tag}, \texttt{gagct}, \texttt{ctagt}\}$



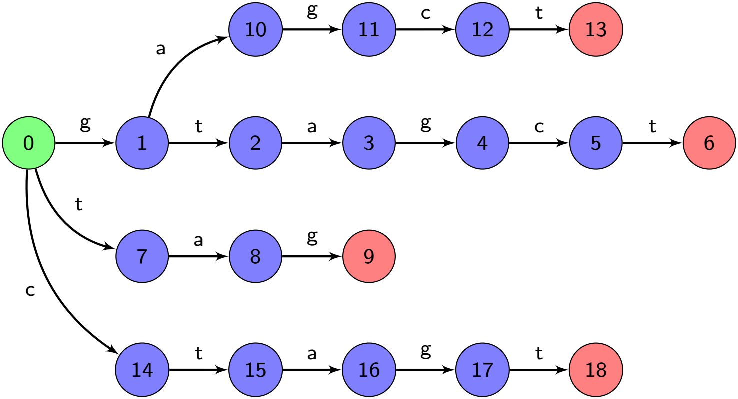
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



Exemple $X = \{\texttt{gtagct}, \texttt{tag}, \texttt{gagct}, \texttt{ctagt}\}$



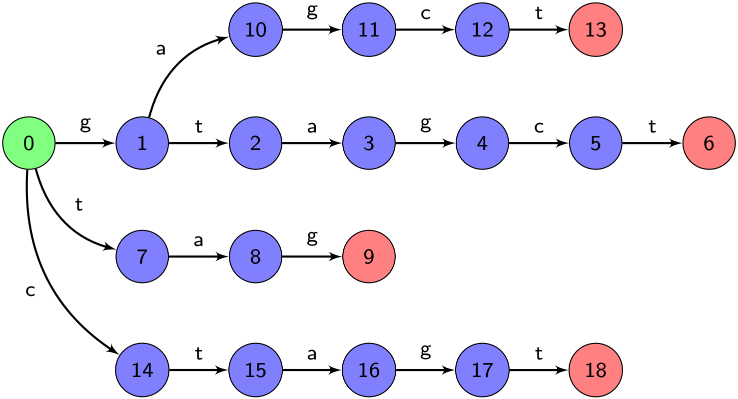
Exemple $X = \{\texttt{gtagct}, \texttt{tag}, \texttt{gagct}, \texttt{ctagt}\}$



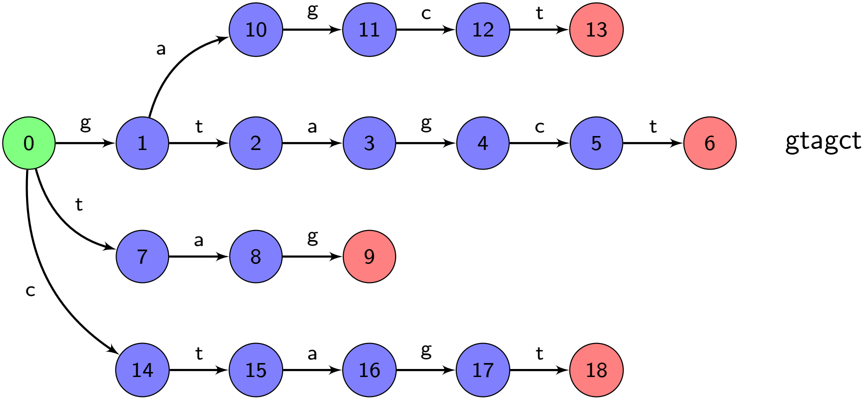
Suite de la construction

- 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 : $\delta(q_0, a) = q_0$ pour $a \in A$ et $a \notin$ *Préf*(X)
- On associe un état suppléant à chaque état : $sup(q) = u$ 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)

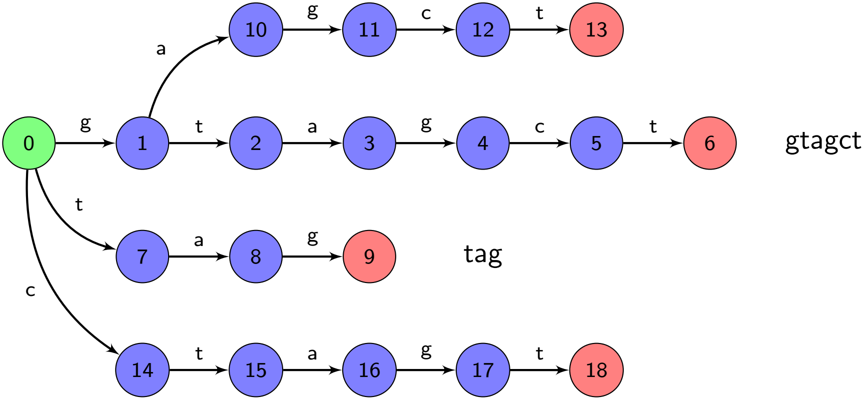
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



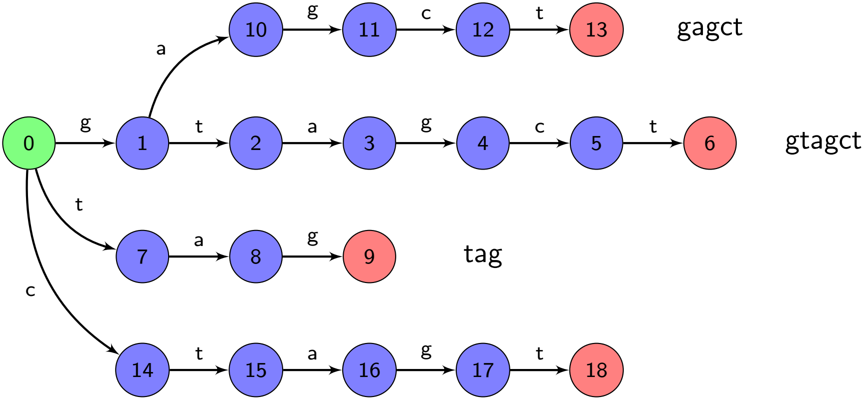
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



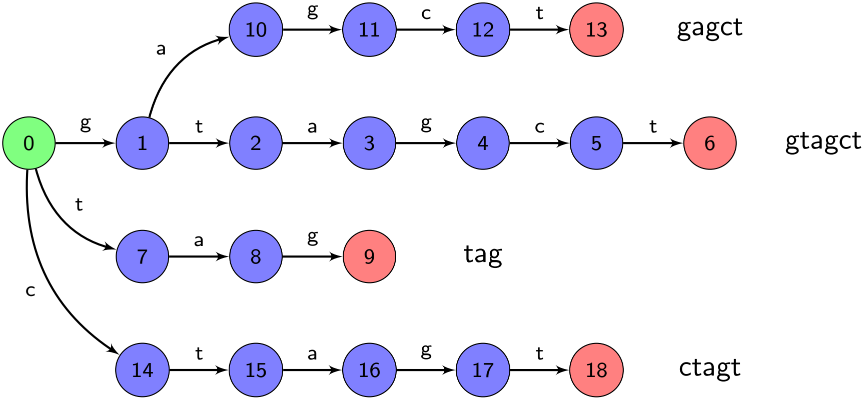
Exemple $X = \{\texttt{gtagct}, \texttt{tag}, \texttt{gagct}, \texttt{ctagt}\}$



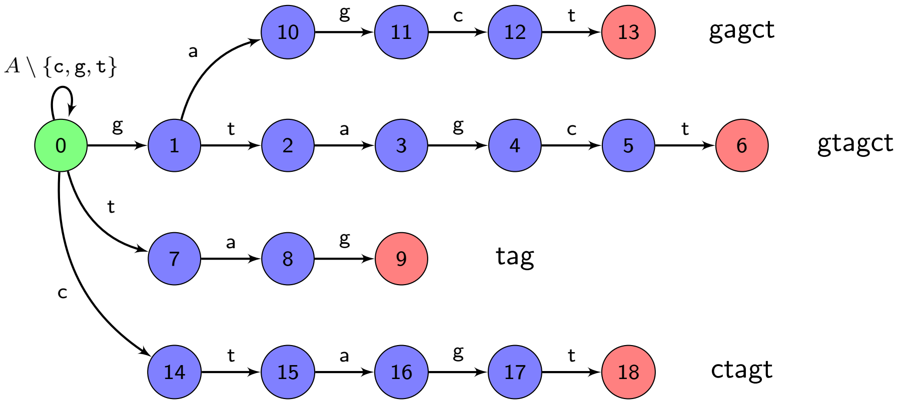
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



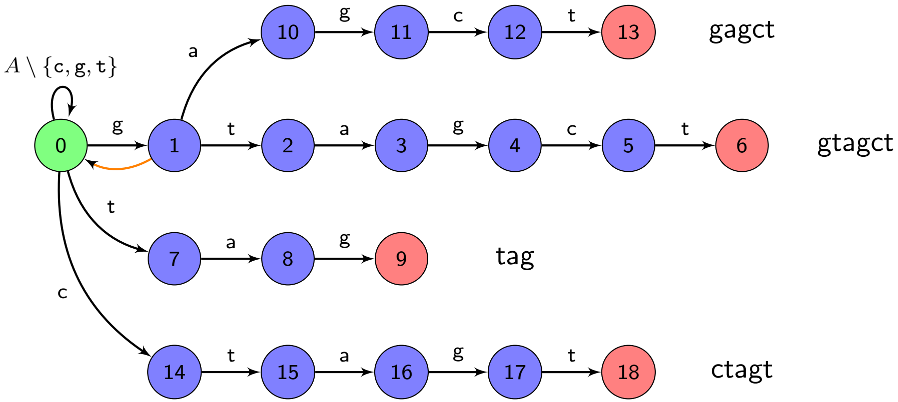
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



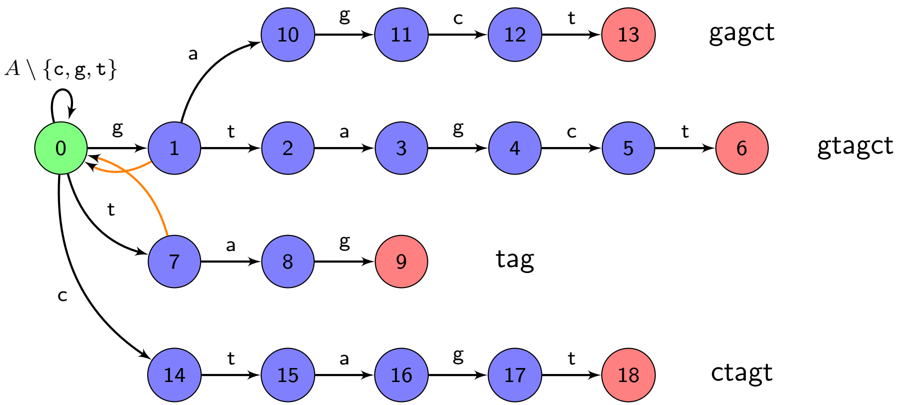
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



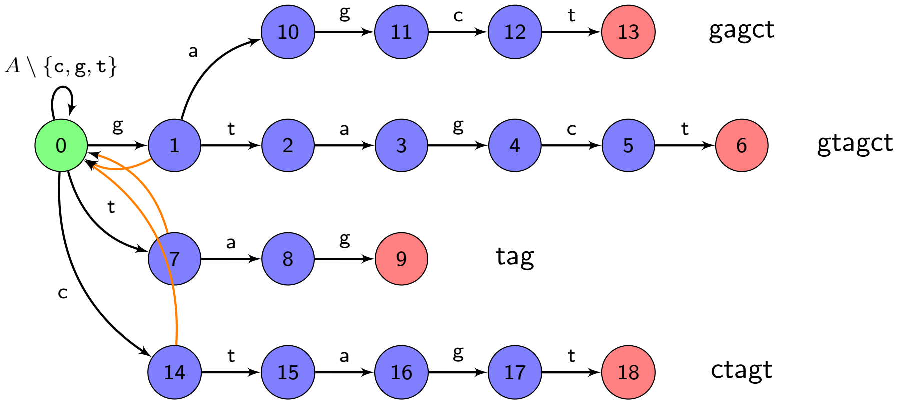
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



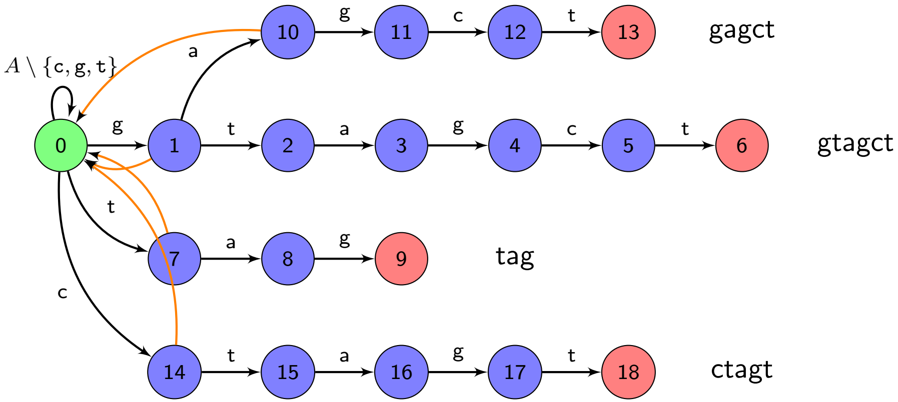
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



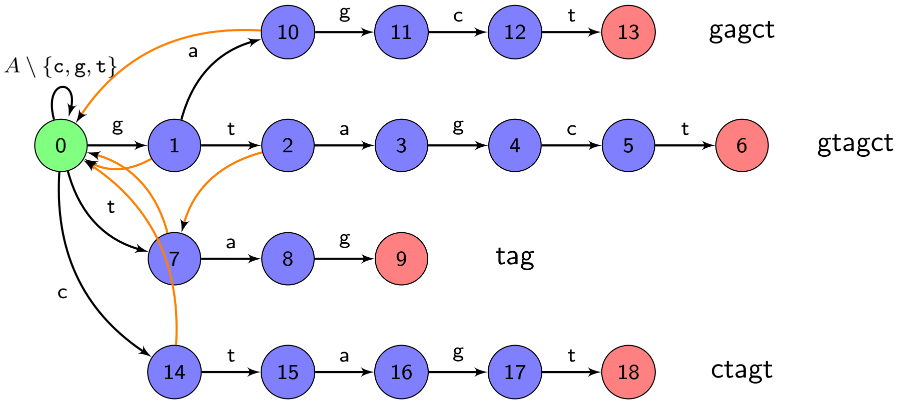
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



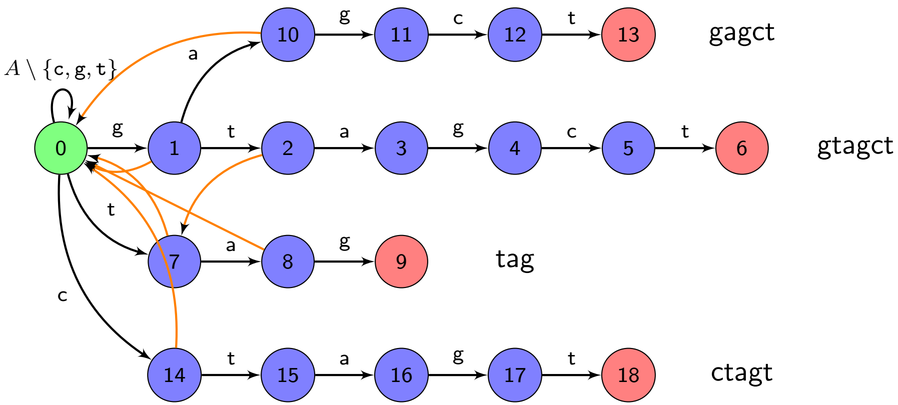
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



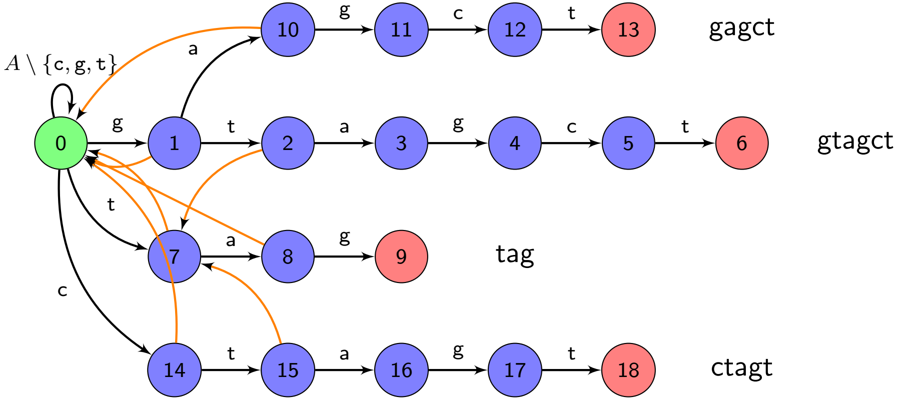
Exemple $X = \{\texttt{gtagct}, \texttt{tag}, \texttt{gagct}, \texttt{ctagt}\}$



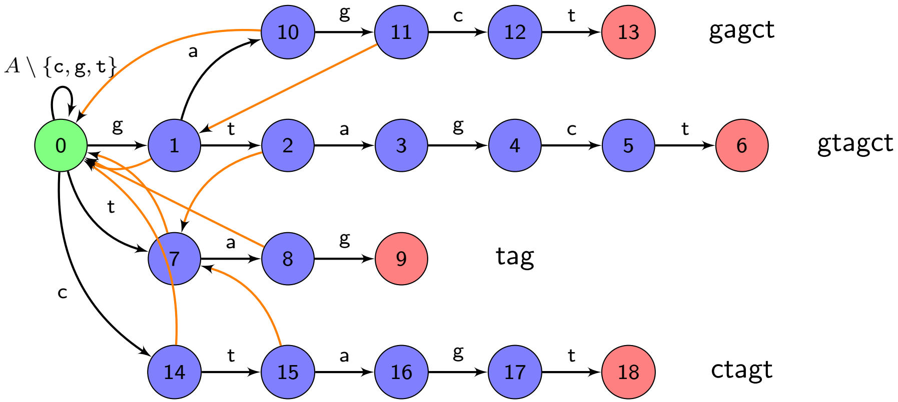
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



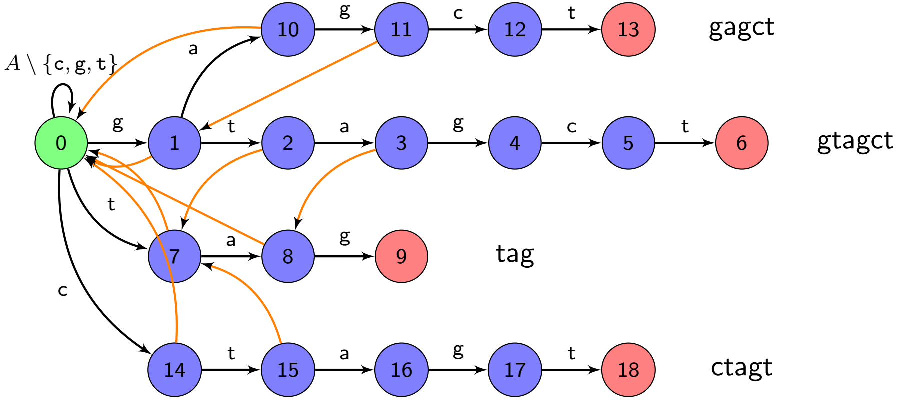
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



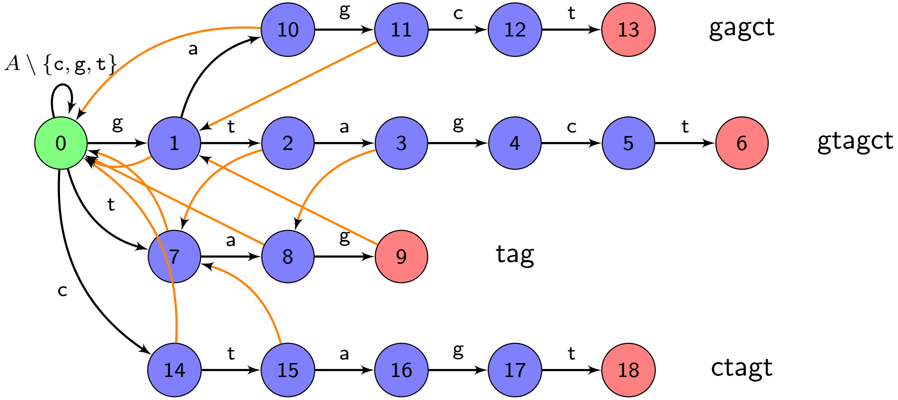
Exemple $X = \{\texttt{gtagct}, \texttt{tag}, \texttt{gagct}, \texttt{ctagt}\}$



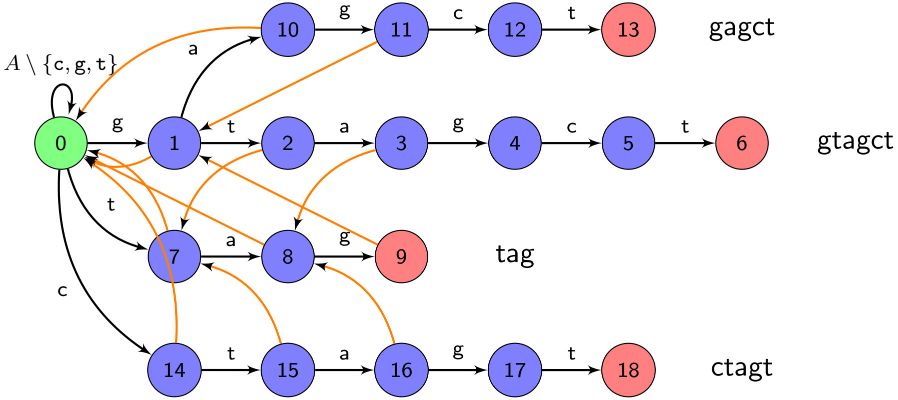
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



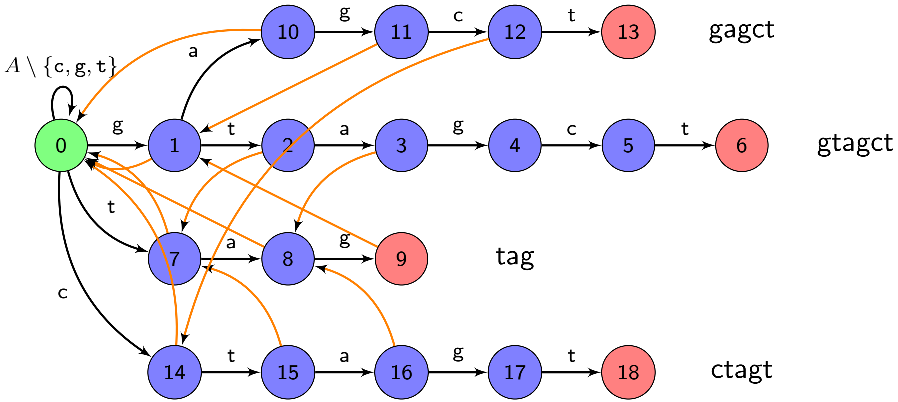
Exemple $X = \{\texttt{gtagct}, \texttt{tag}, \texttt{gagct}, \texttt{ctagt}\}$



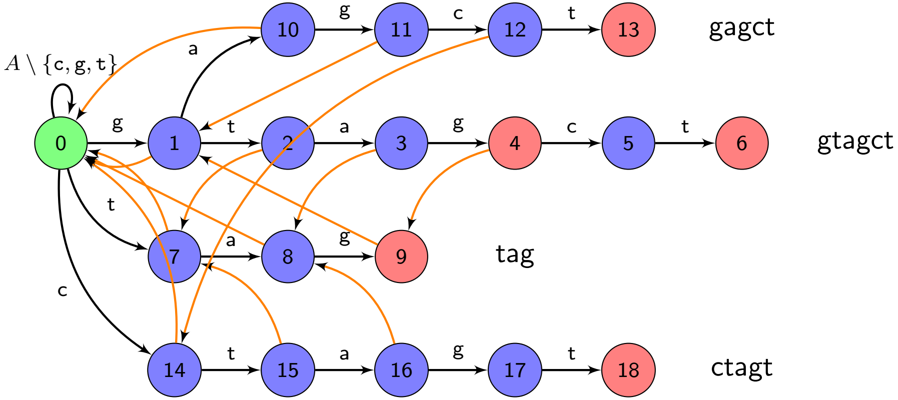
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



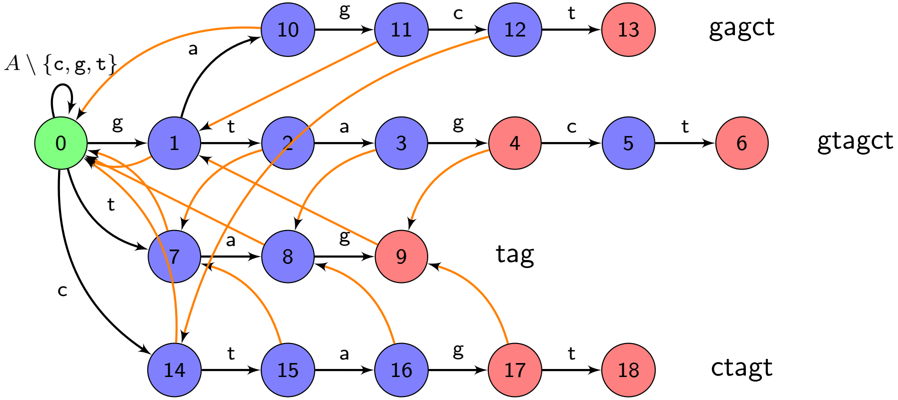
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



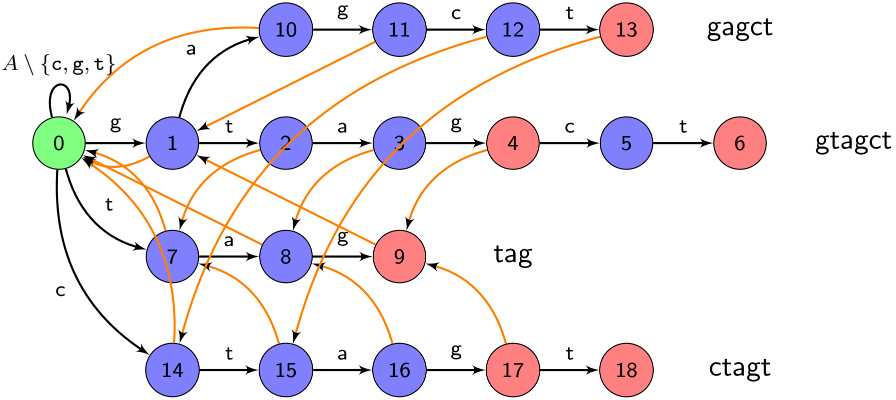
Exemple $X = \{\texttt{gtagct}, \texttt{tag}, \texttt{gagct}, \texttt{ctagt}\}$



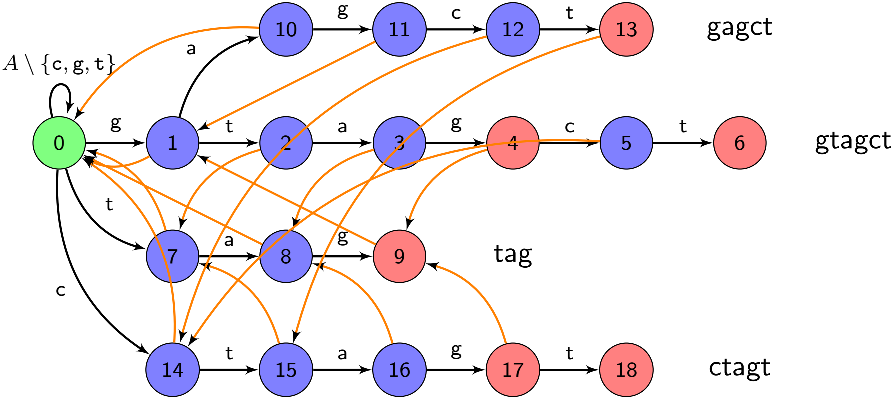
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



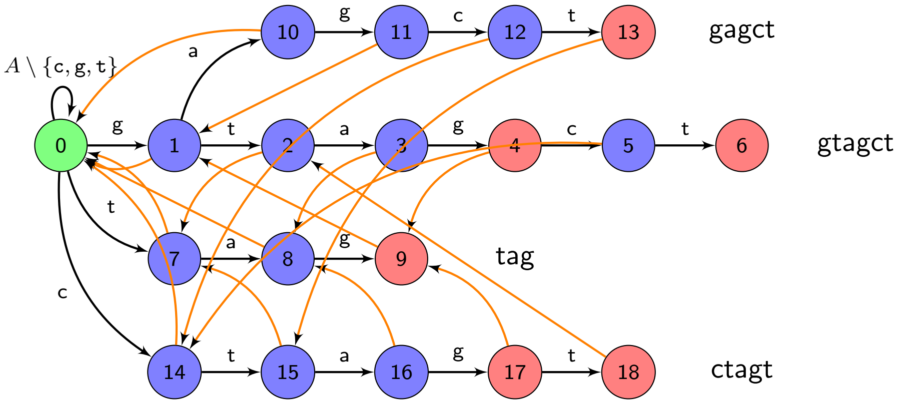
Exemple $X = \{\mathtt{gtagct}, \mathtt{tag}, \mathtt{gagct}, \mathtt{ctagt}\}$



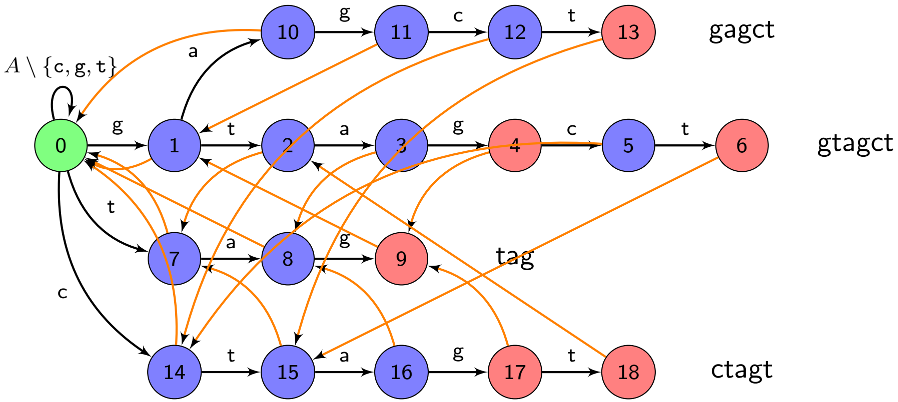
Exemple $X = \{\texttt{gtagct}, \texttt{tag}, \texttt{gagct}, \texttt{ctagt}\}$



Exemple $X = \{\texttt{gtagct}, \texttt{tag}, \texttt{gagct}, \texttt{ctagt}\}$



Exemple $X = \{\texttt{gtagct}, \texttt{tag}, \texttt{gagct}, \texttt{ctagt}\}$



Construction de la fonction de suppléance

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

Algorithmes

Pré-AC(X, k)

créer l'état q_0
pour $i \leftarrow 0$ **à** $k-1$ **faire**
  $\textsc{Entrer}(X[i], q_0)$
pour $a \in A$ **faire**
  si $\delta(q_0, a)$ n'est pas définie **alors**
    $\delta(q_0, a) \leftarrow q_0$
$\textsc{Compléter}(q_0)$
Retourner q_0

Algorithmes

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

Algorithmes

Compléter(e)

```
f ← file vide
ℓ ← liste des transitions (e, a, p) telles que p ≠ e
tantque ℓ est non vide faire
    (r, a, p) ← Premier(ℓ)
    ℓ ← Suivant(ℓ)
    Enfiler(f, p)
    sup(p) ← e
tantque f est non vide faire
    r ← Défiler(f)
    ℓ ← liste des transitions (r, a, p)
    tantque ℓ est non vide faire
        (r, a, p) ← Premier(ℓ)
        ℓ ← Suivant(ℓ)
        Enfiler(f, p)
        s ← sup(r)
        tantque δ(s, a) est non définie faire
            s ← sup(s)
        sup(p) ← δ(s, a)
        sortie(e) ← sortie(e) ∪ sortie(sup(p))
```

Complexité de la phase de prétraitement

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

Exemple

$y = \texttt{ctgagtagctag}$

Algorithmes

AC(X, k, y, n)

```
e ← PRE-AC(X, k)
pour j ← 0 à n − 1 faire
    tantque δ(e, a) est non définie faire
        e ← sup(e)
    e ← δ(e, y[j])
    si sortie(e) ≠ ∅ alors
        reporter une occurrence des éléments de sortie(e)
```

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 temps $O(|X| + n)$ (sur un alphabet constant).

Référence

Efficient string matching : an aid to bibliographic search
A. Aho et M.J. Corasick
Communications of the ACM **18**(6) (1975) 333–340.