RECHERCHE DE MOTIFS

Pb On cherche un mot x, appelé le motif, de taille m, dans un texte t de taille n.

On écrit $x = x_1 x_2 \dots x_m$ et $t = t_1 \dots t_n$.

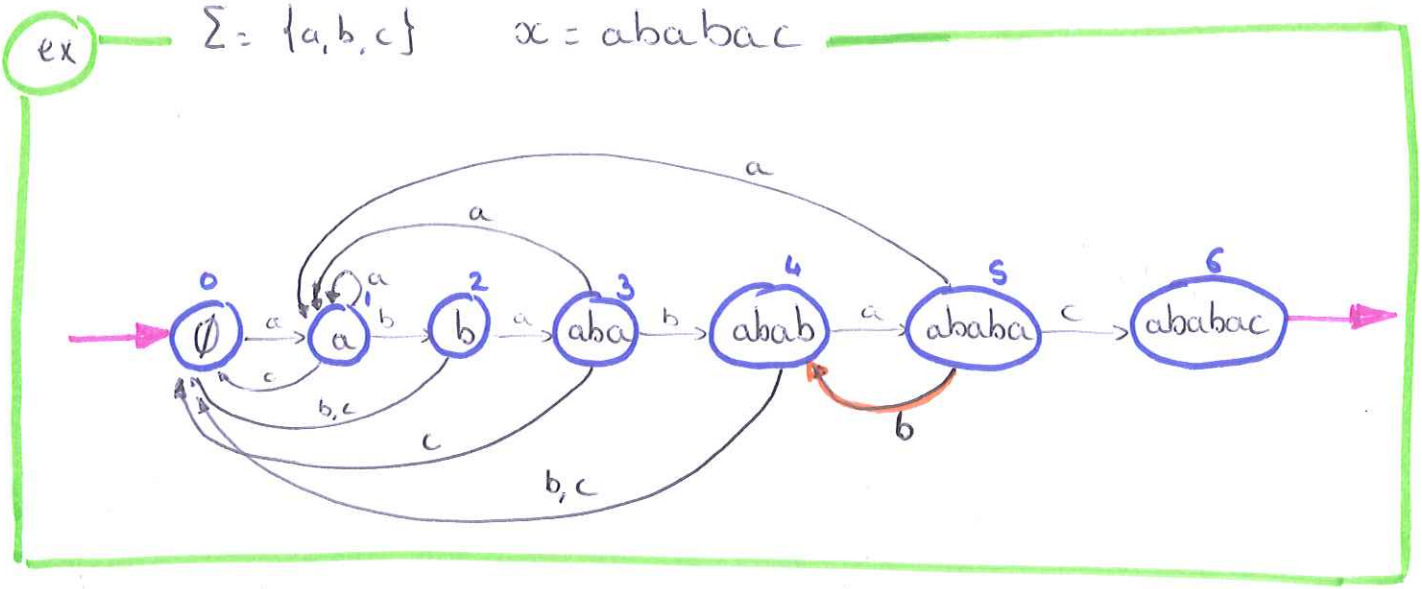
Plus précisément on voudra renvoyer tous les indices d'occurences de x dans t.

Automate du motif

aussi appelé automate de Simon.

On cherche à construire un automate qui reconnait le motif, (ie un automate A tel que $\mathcal{L}(A) = \{x\}$) et qui est complet et déterministe.

ex $\Sigma = \{a, b, c\}$ $x = abababac$



L'idée c'est de garder ce qu'on vient de lire même si on ne peut plus avancer dans le mot, parce que les j dernières lettres qu'on vient de lire constituent peut-être les j premières du motif.

ex [Après avoir lu ababa, si on lit un b, les quatres dernières lettres lues sont a , qui est le début du motif. C'est ce que traduit

A chaque cran de lecture $q \in [0..m]$, c'est-à-dire après avoir lu $x_1 \cdots x_q$, et pour chaque lettre $a \in \Sigma$ on se demande si la fin de $x_1 \cdots x_q a$ est le début du motif, plus exactement on cherche le plus long suffixe de $x_1 \cdots x_q a$ qui soit préfixe de x. C'est l'idée de l'algorithme suivant :

```
transitions_Simon (x = x_1 ··· x_m, m)
    Soit δ une matrice indicé par [0..m] × Σ, initialisée à 0
    Pour q allant de 0 à m
        Pour a parcourant Σ
            k ← 1
            tant que δ(q, a) = 0 et k ≤ q
                si x_k ··· x_q a = x_1 ··· x_{q-k+2}
                alors δ(q, a) = q - k + 2
                sinon k ← k + 1
    retourner δ.
```

Cet algo fournit sous la forme d'une matrice les transitions de l'automate du motif définit par :

$$\mathcal{A}_x = ([0..m], \Sigma, \delta, 0, \{m\})$$

implicitement dû le test $x_k \cdots x_q a = x_1 \cdots x_{q-k+2}$

Au pire des cas la complexité est $\sum_{q=0}^{m} \sum_{a \in \Sigma} \sum_{k=1}^{q} \sum_{i=1}^{q-k+2} 1.$

soit $|\Sigma| \sum_{q=0}^{m} \left(\underbrace{\sum_{k=1}^{q} q-k}_{\sum_{k=1}^{q} k \sim q^2} + q^2 \right) \sim |\Sigma| \sum_{q=0}^{m} q^2 \sim |\Sigma| m^3$

D'où une complexité en $O(|\Sigma| m^3)$

Algo de Simon

L'idée est simple : on va juste faire tourner le super automate qu'on vient de construire.

```
algo_de_Simon (t, x)
    q ← 0 ; δ = transitions_Simon(x)
    Pour i allant de 1 à n
        q ← δ(q, t_i)
        si q = m
        alors imprimer "position %d", i-m
```

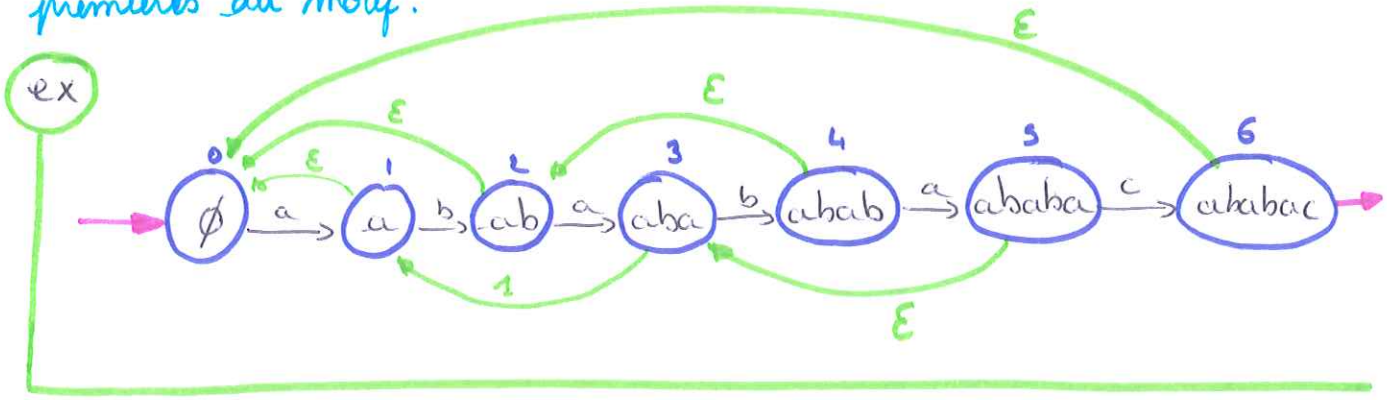
L'essentiel de la complexité tient au prétraitemt en $O(|\Sigma| m^3)$, tandis que le parcours du texte est seulemt en $O(n)$.

D'où une complexité temporelle en $\boxed{O(|\Sigma| m^3 + n)}$

et une complexité spatiale en $\boxed{O(|\Sigma| m)}$ (du fait de δ.)

Automate du motif bis

On s'autorise maintenant les ε-transitions.
L'idée est alors de regarder si les j dernières lettres lues (dans un état donné, sans compter ce qu'on va lire juste après) sont les j premières du motif.



L'automate ne compote alors que les transitions liées à la lecture directe (les $\longrightarrow$) et les ε-transitions. On donne donc ci-dessous un algo calculant ces ε-transitions, sachant qu'on en a une et une seule partant de chaque état.

ε-transitions $(x = x_1 \cdots x_m, m)$

Π un tableau indicé de 1 à m.
$\Pi(1) \leftarrow 0$
$k \leftarrow 0$
Pour q allant de 2 à m
  tant que $x_{k+1} \neq x_q$ et $k > 0$
    $k \leftarrow \Pi(k)$
  si $x_{k+1} = x_q$
  alors $k \leftarrow k+1$
  $\Pi(q) \leftarrow k$
retourner Π.

on remonte dans le motif jusqu'à ce que la lettre qu'on vient de lire à savoir x_q, coincide avec celle qui suit le préfixe envisagé $(x_1 \cdots x_k)$ à savoir x_{k+1}.

si on s'est arrêté parce que ça coincidait on a bien $x_1 \cdots x_{k+1} = x_{q-k} \cdots x_q$ et donc $\Pi(q) = k+1$

en revanche si on s'est arrêté parce que $k=0$, c'est qu'aucun préfixe de x n'est suffixe de $x_1 \cdots x_q$, et donc $\Pi(q) = 0$.

Rq Si $\Pi(i) = j$, j est la taille du plus long suffixe strict de $x_1 \cdots x_i$ qui soit un préfixe de x. En particulier on a $x_{i-j} \cdots x_i = x_1 \cdots x_j$
($x_{i-j} \cdots x_i$: suffixe de $x_1 \cdots x_i$; $x_1 \cdots x_j$: préfixe de x.)

ex avec x = abababc $\quad \Pi(1) = 0$

$k=0$	$q=2$	$a = x_1 \neq x_2 = b$	donc $\Pi(2) = 0$
$k=0$	$q=3$	$a = x_1 = x_3 = a$	donc $k+1$, $\Pi(3) = 1$ et $k=1$
$k=1$	$q=4$	$b = x_2 = x_4 = b$	donc $k+1$, $\Pi(4) = 2$
$k=2$	$q=5$	$a = x_3 = x_5 = a$	$k+1$, $\Pi(5) = 3$
$k=3$	$q=6$	$b = x_4 \neq x_6 = c$	de $k \leftarrow \Pi(3) = 1$
$k=1$	$q=6$	$b = x_2 \neq x_6 = c$	de $k \leftarrow \Pi(1) = 0$, $\Pi(6) = 0$

Algo KMP

Knuth, Morris et Pratt.

```
KMP ( x = x1 — xm , t = t1 — tn )
    π ← E_transitions (x)
    k ← 0
    Pour i allant de 1 à n
        tant que k > 0 et x_{k+1} ≠ ti
            k ← π(k)
        si x_{k+1} = ti
        alors k ← k+1
        si k = m
        alors imprimer "position %d", i-m ; k ← π(k)
```

i = avancement dans le texte
k = ——— le motif

Complexité en nombre de comparaison de lettre.

↳ de E_transitions

Pour compter le nombre de tests on compte le nombre de variations de k.

$N_{k\nearrow}$ = nombre d'incrementation de k

$N_{k\searrow}$ = nombre de diminution de k.

Puisque $k \geq 0$ (au début) à la fin on a $N_{k\nearrow} \geq N_{k\searrow}$.

Or k ne peut être incrémenté qu'une seule fois par boucle

donc $N_{k\nearrow} \leq m$, d'où $N_{k\searrow} \leq m$

Or à q fixé (ie dans un corps de boucle), on fait deux tests de plus que de diminution de k

(test, diminution, test diminue, test sortie du tant que, test du si)

Donc Nb test $\leq N_{k\nearrow} + 2m \leq 3m$ d'où une complexité de E_transition en $O(m)$

↳ de KMP

Ici on procède de même, en comptant le nombre de vacillations de k.

On a encore $Nb_{\searrow} \leq Nb_{\nearrow} \leq n$ (nombre de boucle).

et à chaque boucle i, $Nb\ test(i) \leq Nb_{\searrow}(i) + 2$

D'où $Nb\ test \leq n + 2n$, comptant aussi celle de E-ta (Pour transit°) on a $Nb\ test \leq 3\,min(m,n)$

Donc la complexité de KMP en temps est $O(m+n)$

de plus ——— en espace est $O(n)$