

Les sujets et corrections sont disponibles à l'adresse WEB
<http://www.di.ens.fr/~poupard/ENSTA>

1 Algorithme naïf de pattern-matching

Un premier algorithme très simple mais peu efficace permettant la recherche de motifs dans une chaîne est décrit ci-dessous :

Algorithm 1 RECH-MOTIF-NAÏVE(P, T)

```

1  for  $s \leftarrow 0$  to  $n - m$  do
2      if DÉCALAGE-VALIDE( $P, T, s$ ) then
3          classer  $s$  parmi les décalages valides

4  DÉCALAGE-VALIDE( $P, T, s$ ) =
5       $\forall j \in [1, m], P[j] = T[s + j]$ 

```

Exercice 1 Implémenter l'algorithme naïf de pattern-matching.

2 Algorithme de Rabin-Karp

Afin d'obtenir un algorithme de complexité bien meilleure, on peut utiliser un astucieux codage des mots sous forme d'entiers. Soit d le nombre de symboles qui composent l'alphabet Σ . On définit les entiers suivants au moyen d'une bijection canonique entre les symboles et l'ensemble $[0, d[$:

- $p = \sum_{j=1}^m P[j]d^{m-j}$
- pour tout $s \in [0, n - m]$, $t_s = \sum_{j=1}^m T[s + j]d^{m-j}$

Les entiers ainsi définis peuvent être vus comme un codage en base d des mots. Notez que le calcul de t_{s+1} se fait simplement à partir de t_s en temps constant au moyen de l'équation de récurrence :

$$t_{s+1} = (t_s - T[s + 1] \times h) \times d + T[s + m + 1]$$

où $h = d^{m-1}$.

L'algorithme de recherche de motif de Rabin et Karp s'écrit alors de la manière suivante :

Algorithm 2 RABIN-KARP(P, T, d)

```

1  calcul de  $h \leftarrow d^{m-1}$ 
2  calcul de  $p$ 
3  calcul de  $t \leftarrow t_0$ 
4  for  $s \leftarrow 0$  to  $n - m - 1$  do
5      if  $p = t$  then classer  $s$  valide
6       $t \leftarrow (t - T[s + 1]h)d + T[s + m + 1]$ 
7  if  $p = t$  then classer  $n - m$  valide

```

Exercice 2 Implémenter cet algorithme. Quelle est sa complexité ?

Exercice 3 Lorsque d devient grand, on préfère l'algorithme suivant. Expliquer pourquoi et le programmer.

Algorithm 3 RABIN-KARP(P, T, d, q)

```

1  calcul de  $h \leftarrow d^{m-1} \bmod q$ 
2  calcul de  $p \leftarrow P \bmod q$ 
3  calcul de  $t \leftarrow T_0 \bmod q$ 
4  for  $s \leftarrow 0$  to  $n - m - 1$  do
5      if  $p = t$  then
6          if DÉCALAGE-VALIDE( $P, T, s$ ) then
7              classer  $s$  valide
8           $t \leftarrow ((t - T[s + 1]h)d + T[s + m + 1]) \bmod q$ 
9  if  $p = t$  then
10     if DÉCALAGE-VALIDE( $P, T, s$ ) then
11         classer  $n - m$  valide

```

3 Pattern-matching à base d'automates finis

Pour tout motif P de longueur m , on peut construire un automate M qui le reconnaisse tel que :

- M ait $m + 1$ états : $\{0, 1, \dots, m\}$;
- 0 soit l'état initial ;
- m soit le seul état final.

Par exemple, avec l'alphabet $\Sigma = \{a, b\}$, l'automate de la figure 1 reconnaît le mot $aabba$ et seulement lui.

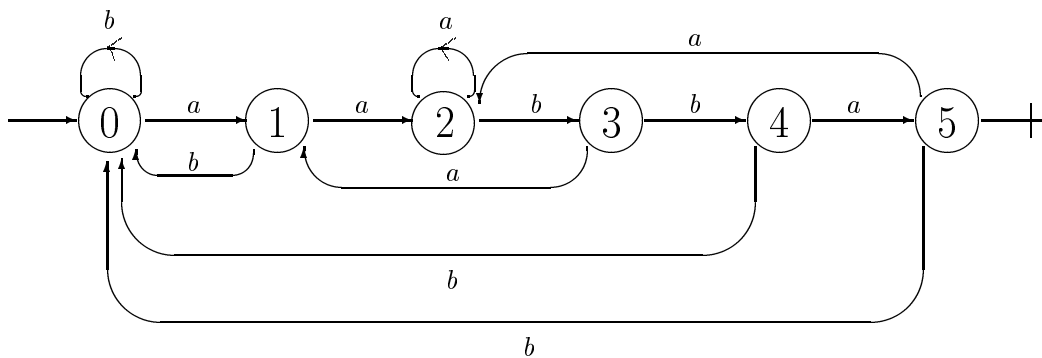


FIG. 1 – Automate de recherche de motif

On en déduit un algorithme très simple, de complexité optimale $\Theta(n)$, permettant de rechercher un motif dans un texte, une fois construit l'automate correspondant au motif :

Algorithm 4 RECH-AUTOMATE(δ, T)

```
1   $q \leftarrow 0$ 
2  for  $s \leftarrow 1$  to  $n$  do
3       $q \leftarrow \delta(q, T[s])$ 
4      if  $q = m$  then
5          classer  $s - m$  valide
```

Calcul de l'automate de recherche

Le problème de la construction de l'automate de recherche n'est cependant pas simple. Il est fabriqué à partir d'un "squelette" linéaire de m transitions correspondant au motif P . La difficulté réside dans la détermination de la fonction de transition δ .

Soit P_q le préfixe de longueur q de P . la fonction de transition δ doit être telle que $\delta(q, x)$ soit égal à la longueur du plus long préfixe de P également suffixe de la concaténation de P_q et x .

Ainsi dans l'exemple de la figure 1, $\delta(3, a)$ est égal à la longueur du plus long préfixe de $aabba$ également suffixe de $aaba$, i.e. égal à 1. De même, $\delta(5, a)$ est égal à la longueur du plus long préfixe de $aabba$ également suffixe de $aabbaa$, i.e. égal à 2 (le mot commun étant aa).

Exercice 4 *Implémenter la construction de l'automate de recherche de motif ainsi que l'algorithme de pattern-matching associé. Analyser la complexité de votre implémentation.*

Application au motif $aabba$:

```
delta(0, a)=1
delta(0, b)=0
delta(1, a)=2
delta(1, b)=0
delta(2, a)=2
delta(2, b)=3
delta(3, a)=1
delta(3, b)=4
delta(4, a)=5
delta(4, b)=0
```