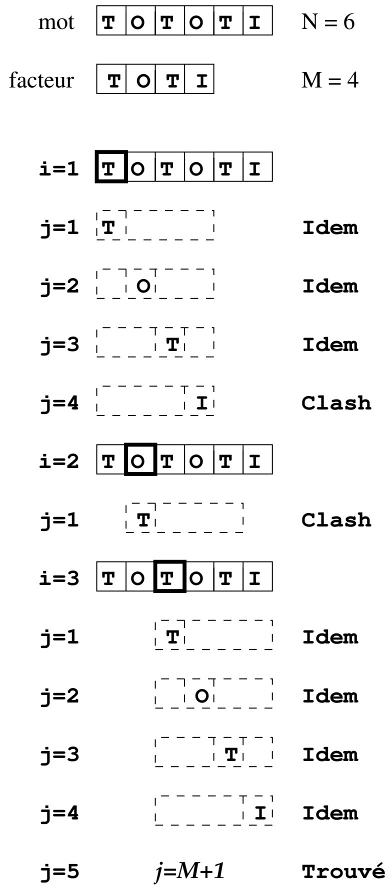
Algorithme(s) de recherche de facteur linéaire(s)

(N = length(mot), M = length(facteur)).




```
i := 1 ; Trouvé := False ;
while (not Trouvé and i <= N) do begin
  j := 1 ; Idem := True ;
  while (Idem and j <= M) do begin
    Idem := (facteur[j] = mot[i+j-1]) ;
    j := j + 1 ;
  end ;
  if Idem then Trouvé := True ;
  i := i + 1 ;
end ;
if Trouvé then writeln('Motif trouvé') ;
```

```
i := 1 ; Trouvé := False ;
while (not Trouvé and i <= N) do begin
  j := 1 ;
  while (facteur[j] = mot[i+j-1] and j <= M) do
    j := j + 1 ;
  if j = M + 1 then Trouvé := True ;
  i := i + 1 ;
end ;
if Trouvé then writeln('Motif trouvé') ;
```

```
i := 0 ;
repeat
  i := i + 1 ;
  j := 1 ;
  while (facteur[j] = mot[i+j-1] and j <= M) do
    j := j + 1 ;
until j = M + 1 or i = N
if j = M + 1 then writeln('Motif trouvé') ;
```

Algorithmes plus efficaces

Knuth, Morris et Pratt Cet algorithme utilise les caractères déjà comparés pour déterminer la prochaine position du facteur à utiliser. A cet effet, une table de taille M est calculée avant la recherche (elle ne dépend que du facteur). L'algorithme est dit *on line*, car il ne revient jamais sur un caractère déjà considéré (même s'il peut considérer plusieurs fois un caractère donné) [Wirth, 1987, p. 53]. Il est de complexité $O(M+N)$, mais le gain qu'il apporte est surtout dans les cas défavorables.

```
Hoola-Hoola j'aime les Hooligans
Hooligan
^^^^^
    Hooligan
    ^
     Hooligan
     ^
      Hooligan
      ^^^^^
          Hooligan
          ^
           Hooligan
           ^  .........
                       Hooligan
                       ^^^^^^^^
```

```
i := 1 ; j := 1 ;
while (j<=M) and (i<=N) do {
  while (mot[i] <> facteur[j]) do
    j := D[j] ;
  i := i+1 ; j := j+1 ;
end ;
if j = M + 1 then
  writeln('Motif trouvé') ;
```

Calcul de D : *précompilation* du modèle.

...	A	A	A	A	A	C				...
	A	A	A	A	A	B				
		A	A	A	A	A	B			

...	A	B	C	A	B	D				...
	A	B	C	A	B	C				
				A	B	C	A	B	C	

...	A	B	C	D	E	A				...
	A	B	C	D	E	F				
						A	B	C	D	...

Boyer-Moore Cet algorithme effectue la comparaison avec le facteur de droite à gauche. Après qu'une différence est trouvée, il calcule un décalage (*shift*), c'est-à-dire le nombre de positions selon lequel le facteur est décalé vers la droite avant qu'une nouvelle comparaison soit tentée. Ce calcul utilise deux tables similaires à celle de KMP, construites avant le début de l'exploration. Cet algorithme est aussi d'ordre M+N, mais il est aussi meilleur dans le cas moyen, contrairement à KMP [Wirth, 1987, p. 59].