

Option Informatique

Centrale 1997 : Algorithme de Knuth-Morris-Pratt

Corrigé de Martine Pages

11 mars 2007

I B 1 Considérons l'indice k initialisé à 0 et incrémenté de une unité à chaque passage à la ligne 10 (c'est à dire retour à la première lettre du motif).

On peut considérer que l'on recommence alors la recherche sur le source privé de ses k premiers caractères (c'est une amorce de la méthode recursive).

En sortie de boucle, les k premiers éléments de source sont donc "rejetés", et si $i > 0$, les éléments de source compris entre k et $k + i - 1 = j - 1$ coïncident avec les éléments de motifs compris entre 0 et $i - 1$.

2 A chaque passage dans la boucle, soit i augmente (mais reste inférieur à la longueur p du motif), soit k augmente (mais reste inférieur à j donc à la longueur n du source). Donc, la boucle ne peut en aucun cas être effectué plus de np fois. (On verra plus loin une amélioration de cette majoration). Lorsque l'on sort de la boucle, soit $i = p$ et les éléments compris entre k et $k + p - 1$ de source correspondent au motif : on a trouvé le motif dans source, soit $i < p$ et $j = n$, donc on a épuisé la source sans avoir trouvé le motif.

3 Fonction `recherche_iterative_brute` :

```
let recherche_iter_brute motif source=  
  let p=vect_length motif and n=vect_length source and i=ref 0 and j=ref 0 in  
  while (!i<p) && (!j<n) do  
    if motif.(!i)=source.(!j) then (i :=!i+1; j :=!j+1)  
      else (j :=!j-!i+1; i :=0) done;  
  if !i=p then true else false;;  
  
let motif=[1;2;1;2;3] and source=[1;2;2;1;2;1;2;3;1;2]  
  in recherche_iter_brute motif source;;  
  
let motif=[1;2;1;2;3] and source=[1;2;2;1;2;1;1;3;1;2]  
  in recherche_iter_brute motif source;;
```

4 Dans le pire des cas, imaginons que l'on regarde à chaque fois les p éléments du motifs, donc à partir des positions 0, 1, ..., $n - p$ dans source, on aura effectué $p.(n - p + 1)$ comparaisons.

Si source = $aaa \dots aac$ et motif = $aaa \dots aab$, on effectuera les $p(n - p)$ comparaisons precedentes, et on continuera en comparant le premier a du motif successivement aux éléments de source situés

aux positions $n - p + k$ pour $k \in [1, p - 1]$ soit encore $p - k$ comparaisons à chaque fois, soit au total : $p(n - p + 1) + \frac{p(p-1)}{2} = \frac{p}{2}(2n - p + 1)$ comparaisons.

- C 1 Note : conformément au texte du problème, la fonction `est_prefixe` (I.C.2) est utilisée dans `recherche_recursive`. Pour compiler en Caml, il est bien sûr nécessaire de l'écrire avant.

Fonction `recherche_recursive` :

```
let rec recherche_recursive motif source=
  if est_prefixe motif source then true
  else if source=[] then false else recherche_recursive motif (tl source);;
```

- 2 Fonction `est_prefixe` :

```
let rec est_prefixe motif source=
  if source=[] then false
  else if motif=[] then true
  else if hd motif<>hd source then false else est_prefixe (tl motif) (tl source);;

let motif=[1;2;3] and source=[1;2;3;4;5] in est_prefixe motif source;;

let motif=[1;2;3] and source=[1;2;4;4;5] in est_prefixe motif source;;
```

- 3 C'est le même résultat qu'à la question 1.B.4 puisque l'algorithme récursif est basé sur le même principe que l'algorithme itératif. Et le pire des cas étudié ci-dessus donne le même résultat.

D Fonction `recherche_KMP` :

```
1 let recherche_KMP motif source tableau=
  let p=vect_length motif and n=vect_length source
  and k= ref 0 and i=ref 0 in
  while (!k<p) && (!i<n) do
    if !k<0 or motif.(!k)=source.(!i) then (i :=!i+1;k :=!k+1)
    else k :=tableau.(!k)
  done;
  if !k<p then false else true;;

let motif=[|1;2;1;2;3|] and source=[|1;2;1;2;1;2;3;1;2|] and tableau=[|-1;0;0;1;2;0|]
in recherche_KMP motif source tableau;;
```

- 2 Le tableau auxiliaire pour le motif "abaabababaabaab" est $(-1, 0, 0, 1, 1, 2, 3, 2, 3, 4, 5, 6, 4, 5)$.

3 Fonction `calcule_tab_aux` :

```
let calcule_tab_aux motif= let p=vect_length motif in
let t=make_vect (p+1) (-1)
in
for j=0 to p-1 do let l=ref 0 in
  for k=1 to j-1 do if prefixe motif (sub_vect motif (j-k+1) k) then l :=k done;
t.(j+1) < -- !l
done;
t;;

let motif=[|1;2;1;1;2;1;2;1;1;2;1;1;2|] in calcule_tab_aux motif;;
```