Mise en œuvre de la recherche de motif par l'algorithme de Boyer-Moore

Julien Reichert

Algorithme de Boyer-Moore

On note `s` le texte, n sa taille, `m` le motif et k sa taille. On recherche cette fois-ci le nombre d'occurrences de `m` dans `s`.

Il s'agit de comparer les caractères de chaque facteur testé dans `s` avec les caractères de `m`, mais **depuis la fin**.

Pour comprendre l'intérêt : si le dernier caractère du facteur de `s` testé n'est pas dans `m`, alors non seulement le facteur de `s` n'est pas `m`, mais les `k-1` suivants non plus, donc le prochain qu'on testera pourra se trouver `k` crans plus loin.

Le problème qui se pose est alors la manière de savoir comment tester ceci rapidement et comment gérer d'autres cas.

On se limite à la version simple ici !

Algorithme de Boyer-Moore

- Calcul initial sur `m` effectué une fois pour toutes.
- Construction d'un tableau de dictionnaires (un tableau de dimension deux serait trop lourd) associé à `m`.
- Indexation du tableau : par l'indice où on en est dans la comparaison des caractères du facteur de `s` avec ceux de `m`
- Dictionnaire à un indice particulier : indexé par des caractères, décalage donnant l'indice du prochain facteur testé en fonction du caractère rencontré.

« *La recherche de facteurs dont vous êtes le héros !* »

Explication sur un exemple

Pour le moment, on ne se concentre que sur `m` (pour que cela ait de l'intérêt, les lettres ne seront pas toutes différentes).

Soit `m = "CARACTERE"`, de lettres distinctes `A`, `C`, `E`, `R` et `T`.

Pour chaque autre lettre, la rencontrer fera décaler le premier indice du prochain facteur testé juste après l'indice où cette lettre aura été rencontrée.

Sinon, soit la lettre correspond au facteur recherché et on continue, soit on décale la plage d'indices du facteur testé du plus petit nombre de crans nécessaire vers la droite pour aligner ce caractère avec le même caractère dans `m` si c'est encore possible (sinon on fait comme si la lettre était absente).

Explication sur un exemple

Illustration d'un cas possible :

Indice s	...	i							$i+7$	$i+8$	...
s	...	s_i	?	?	?	?	?	?	T	E	...
m		C	A	R	A	C	T	E	R	E	
Indice m		0							7	8	

$\longrightarrow$

Indice s	...	$i+2$							$i+9$	$i+10$	...
s	...	s_{i+2}	?	?	?	?	**T**	E	?	s_{10}	...
m		C	A	R	A	C	**T**	E	R	E	
Indice m		0							7	8	

Explication sur un exemple

Illustration d'un autre cas possible :

Indice s	...	i					$i+5$	$i+6$	$i+7$	$i+8$	...
s	...	s_i	?	?	?	?	E	E	R	E	...
m		C	A	R	A	C	T	E	R	E	
Indice m		0							7	8	

$\longrightarrow$

Indice s	...	$i+6$								$i+14$	...
s	**E**	(E)	(R)	(E)	?	?	?	?	?	s_{14}	...
m		C	A	R	A	C	T	E	R	E	
Indice m		0								8	

(optimisable)

Méthode de construction des dictionnaires

Attention : Construction indice par indice depuis le premier indice, même si l'utilisation se fera depuis le dernier indice.

Valeur associée à une clé : de combien l'indice de début du facteur de `s` augmentera pour le prochain tour de boucle (la boucle extérieure sera en particulier conditionnelle). Si la lettre lue n'est pas une clé, le décalage est égal à l'indice du dictionnaire actuel plus un.

Cas particulier : à chaque indice, la valeur 0 est réservée à la lettre qu'il fallait reconnaître dans `m`, pour faciliter les calculs par la suite.

Méthode de construction du dictionnaire

- Indice 0 : une seule clé, la première lettre du mot.
- D'un indice à l'indice suivant, toutes les clés existantes voient leur valeur augmentée d'un.
- Puis la lettre correspondante du mot est associée à la valeur 0, en tant que nouvelle clé ou en remplacement de la valeur précédente.

Explication sur un exemple

Le décalage en fonction de la lettre rencontrée à un certain moment de l'étude du facteur est donné par le tableau suivant :

En bleu figurent les nombres qui pourraient être optimisés dans une version plus avancée de l'algorithme.

Clés des dictionnaires :	'A'	'C'	'E'	'R'	'T'
Dictionnaire à l'indice 0 ('C')		WIN			
Dictionnaire à l'indice 1 ('A')	OK	+1			
Dictionnaire à l'indice 2 ('R')	+1	+2		OK	
Dictionnaire à l'indice 3 ('A')	OK	+3		+1	
Dictionnaire à l'indice 4 ('C')	+1	OK		+2	
Dictionnaire à l'indice 5 ('T')	+2	+1		+3	OK
Dictionnaire à l'indice 6 ('E')	+3	+2	OK	+4	+1
Dictionnaire à l'indice 7 ('R')	+4	+3	+1	OK	+2
Dictionnaire à l'indice 8 ('E')	+5	+4	OK	+1	+3

L'algorithme

On suppose le tableau de dictionnaires construit par `precalcule`.

```
fonction boyer_moore(s, m) {
  tab <- precalcule(m);
  n <- taille(s); k <- taille(m); i <- 0; reponse <- 0;
  tant que i < n - k + 1 {
    j <- k-1;
    tant que j > -1 et s[i+j] = m[j] décrémenter j;
    si j = -1 alors {
      incrémenter reponse; incrémenter i;
    }
    sinon si s[i+j] est une clé de tab[j] alors
      augmenter i de tab[j][s[i+j]];
    sinon augmenter i de j+1;
  }
  renvoyer reponse;
}
```

Mise en œuvre

```
s = "COROCTER|E| CARACTERIEL CARACTERE"

m = "CARACTER|E|"

reponse = 0

i = 0

j = 8
```

Correspondance : on reste dans la boucle.

Mise en œuvre

```
s = "COROCTE|R|E CARACTERIEL CARACTERE"

m = "CARACTE|R|E"

reponse = 0

i = 0

j = 7
```

Correspondance : on reste dans la boucle.

Mise en œuvre

```
s = "COROCT|E|RE CARACTERIEL CARACTERE"

m = "CARACT|E|RE"

reponse = 0

i = 0

j = 6
```

Correspondance : on reste dans la boucle.

Mise en œuvre

```
s = "COROC|T|ERE CARACTERIEL CARACTERE"

m = "CARAC|T|ERE"

reponse = 0

i = 0

j = 5
```

Correspondance : on reste dans la boucle.

Mise en œuvre

```
s = "CORO|C|TERE CARACTERIEL CARACTERE"

m = "CARA|C|TERE"

reponse = 0

i = 0

j = 4
```

Correspondance : on reste dans la boucle.

Mise en œuvre

```
s = "COR|O|CTERE CARACTERIEL CARACTERE"

m = "CAR|A|CTERE"

reponse = 0

i = 0

j = 3
```

Échec de la correspondance, sortie de la boucle. Décalage : 4 crans.

Mise en œuvre

```
s = "COROCTERE CA|R|ACTERIEL CARACTERE"

m = "CARACTER|E|"

reponse = 0

i = 4

j = 8
```

Échec de la correspondance, sortie de la boucle. Décalage : 1 cran.

Mise en œuvre

```
s = "COROCTERE CAR|A|CTERIEL CARACTERE"

m = "CARACTER|E|"

reponse = 0

i = 5

j = 8
```

Échec de la correspondance, sortie de la boucle. Décalage : 5 crans.

Mise en œuvre

```
s = "COROCTERE CARACTER|I|EL CARACTERE"

m = "CARACTER|E|"

reponse = 0

i = 10

j = 8
```

Échec de la correspondance, sortie de la boucle. Décalage : 9 crans.

Mise en œuvre

```
s = "COROCTERE CARACTERIEL CARAC|T|ERE"

m = "CARACTER|E|"

reponse = 0

i = 19

j = 8
```

Échec de la correspondance, sortie de la boucle. Décalage : 3 crans.

Mise en œuvre

```
s = "COROCTERE CARACTERIEL CARACTER|E|"

m = "CARACTER|E|"

reponse = 0

i = 22

j = 8
```

Correspondance : on reste dans la boucle.

Mise en œuvre

`s = "COROCTERE CARACTERIEL CARACTERE"`

`m = "CARACTERE"`

Huit autres tests plus tard, la correspondance est trouvée !

Bilan

On quitte la boucle conditionnelle une fois que `i = 23` (on avait `n = 31` et `k = 9`).

Nombre de calculs effectués :

- 31 ajouts dans un dictionnaire
- Tout ce qui tourne autour des structures de contrôle.

Nombre de comparaisons entre caractères effectuées : 19.

Bilan

L'algorithme naïf aurait fait 42 comparaisons de caractères. La rentabilité du précalcul est d'autant plus importante que `s` est grande et `m` assez petite.

Une optimisation de Boyer-Moore aurait permis de décaler bien plus après l'échec sur la lettre O, mais en fin de compte le nombre total de comparaisons n'aurait diminué que d'un, pour une plus grande difficulté de construction.

La raison pour laquelle on ne décale que d'un cran après la réussite est que `m` peut tout à fait être par exemple `"AAAA"`, mais là aussi pour un motif arbitraire le décalage est optimisable.