

1. Position du problème

### Définitions et notations

On s'intéresse au problème de la recherche d'une chaîne de caractères appelée motif dans une autre chaîne de caractères appelée texte. Plus précisement, on veut lister toutes les occurences (par leur position) du motif dans le texte. On notera :

1. Position du problème

### Définitions et notations

On s'intéresse au problème de la recherche d'une chaîne de caractères appelée motif dans une autre chaîne de caractères appelée texte. Plus précisement, on veut lister toutes les occurences (par leur position) du motif dans le texte. On notera :

• m le motif et  $l_m$  sa longueur

1. Position du problème

#### Définitions et notations

On s'intéresse au problème de la recherche d'une chaîne de caractères appelée motif dans une autre chaîne de caractères appelée texte. Plus précisement, on veut lister toutes les occurences (par leur position) du motif dans le texte. On notera :

- ullet m le motif et  $l_m$  sa longueur
- t le texte et  $l_t$  sa longueur

1. Position du problème

#### Définitions et notations

On s'intéresse au problème de la recherche d'une chaîne de caractères appelée motif dans une autre chaîne de caractères appelée texte. Plus précisement, on veut lister toutes les occurences (par leur position) du motif dans le texte.

- On notera :
  - ullet m le motif et  $l_m$  sa longueur
  - t le texte et  $l_t$  sa longueur

D'autre part, parfois le problème se réduira à déterminer si m est présent ou non dans t, ou encore on cherchera uniquement la première occurence.

1. Position du problème

#### Définitions et notations

On s'intéresse au problème de la recherche d'une chaîne de caractères appelée motif dans une autre chaîne de caractères appelée texte. Plus précisement, on veut lister toutes les occurences (par leur position) du motif dans le texte.

#### On notera:

- m le motif et  $l_m$  sa longueur
- ullet t le texte et  $l_t$  sa longueur

D'autre part, parfois le problème se réduira à déterminer si m est présent ou non dans t, ou encore on cherchera uniquement la première occurence.

### Exemple

La recherche du motif m=exe ( $l_m = 3$ ) dans le texte t =un excellent exemple et un exercice extraordinaire ( $l_t = 50$ ) doit produire la liste d'occurrences : [13; 27].

1. Position du problème

#### Définitions et notations

On s'intéresse au problème de la recherche d'une chaîne de caractères appelée motif dans une autre chaîne de caractères appelée texte. Plus précisement, on veut lister toutes les occurences (par leur position) du motif dans le texte. On notera :

- m le motif et  $l_m$  sa longueur
- ullet t le texte et  $l_t$  sa longueur

D'autre part, parfois le problème se réduira à déterminer si m est présent ou non dans t, ou encore on cherchera uniquement la première occurence.

### Exemple

La recherche du motif m=exe ( $l_m = 3$ ) dans le texte t =un excellent exemple et un exercice extraordinaire ( $l_t = 50$ ) doit produire la liste d'occurrences : [13; 27].

 $\underset{\scriptscriptstyle{0}}{\text{un}}_{\sqcup} \text{excellent}_{\underset{\scriptscriptstyle{13}}{\text{exemple}}} \text{exemple}_{\sqcup} \text{et}_{\sqcup} \text{un}_{\underset{\scriptscriptstyle{27}}{\text{exercice}}} \text{extraordinaire}$ 

2. Recherche naïve

#### Recherche naïve

Pour recherche si un motif m se trouve dans un texte t, on peut :

• parcourir chaque caractère de t jusqu'à celui d'indice ? inclus (indice de la dernière occurrence possible) :

indice dans le texte 0 indice dans le motif

	- /		
0		?	$l_t - 1$
		0	$l_m-1$

Pour recherche si un motif m se trouve dans un texte t, on peut :

• parcourir chaque caractère de t jusqu'à celui d'indice  $l_t-l_m$  inclus (indice de la dernière occurrence possible) :

indice dans le texte indice dans le motif

, , ,	3 0 0 0 1 0 1									
0		$l_t - l_m$	$l_t - 1$							
		0	$l_m-1$							

Pour recherche si un motif m se trouve dans un texte t, on peut :

• parcourir chaque caractère de t jusqu'à celui d'indice  $l_t - l_m$  inclus (indice de la dernière occurrence possible) : indice dans le texte  $0 \ \ldots \ l_t - l_m \ l_t - 1$ 

indice dans le motif

	0	 $l_t - l_m$	$l_t - 1$
,		0	$l_m-1$

 $oldsymbol{2}$  si le caractère correspond au premier caractère du motif m, alors on avance dans le motif tant que les caractères coïncident.

Pour recherche si un motif m se trouve dans un texte t, on peut :

• parcourir chaque caractère de t jusqu'à celui d'indice  $l_t - l_m$  inclus (indice de la dernière occurrence possible) : indice dans le texte  $0 \ \ldots \ l_t - l_m \ l_t - 1$ 

indice dans le motif

0	 $l_t - l_m$	$l_t - 1$
•	0	$l_m-1$

- $oldsymbol{2}$  si le caractère correspond au premier caractère du motif m, alors on avance dans le motif tant que les caractères coı̈ncident.
- ullet si on atteint la fin du motif, alors m se trouve dans t. Sinon on passe au caractère suivant de t.

Pour recherche si un motif m se trouve dans un texte t, on peut :

• parcourir chaque caractère de t jusqu'à celui d'indice  $l_t - l_m$  inclus (indice de la dernière occurrence possible) : indice dans le texte  $0 \ \ldots \ l_t - l_m \ l_t - 1$ 

indice dans le motif

Λ.	 1 1	7 1
U	 $\iota_t - \iota_m$	$\iota_t - 1$
	0	$l_m-1$

- $oldsymbol{2}$  si le caractère correspond au premier caractère du motif m, alors on avance dans le motif tant que les caractères coı̈ncident.
- ullet si on atteint la fin du motif, alors m se trouve dans t. Sinon on passe au caractère suivant de t.

### Exemple

Visualisation en ligne du fonctionnement de l'algorithme



## Implémentation en OCaml

On renvoie la *liste* de toutes les occurrences : naive : string -> string -> int list

## Implémentation en OCaml

```
On renvoie la liste de toutes les occurrences : naive : string -> string ->
 int list
   let naive motif texte =
      let lm = String.length motif
                                      in
      let lt = String.length texte in
      let occ = ref [] in
      for it=0 to lt-lm do
        let im = ref 0 in
        while (!im<lm && motif.[!im]=texte.[it+ !im]) do
          im := !im +1;
        done:
        if (!im=lm) then occ := it::!occ
10
        done;
11
      !occ;;
12
```



## Implémentation en OCaml

Dans le cas où on teste simplement la présence, on peut provoquer une sortie anticipée de la boucle for à l'aide d'une exception.

### Implémentation en OCaml

Dans le cas où on teste simplement la présence, on peut provoquer une sortie anticipée de la boucle for à l'aide d'une exception.

```
let present motif texte =
      let lm = String.length motif in
      let lt = String.length texte in
      try
      for it=0 to lt-lm do
        let im = ref 0 in
        while (!im<lm && motif.[!im]=texte.[it+ !im]) do
          im := !im +1:
        done:
        if (!im=lm) then raise Exit (* Exit est prédéfinie *)
10
      done;
11
      false
12
      with Exit -> true;;
13
```



## Coût de la recherche simple

En notant  $l_m$  la longueur du motif et  $l_t$  celle de la chaine :



## Coût de la recherche simple

En notant  $l_m$  la longueur du motif et  $l_t$  celle de la chaine :

ullet La boucle for est parcourue au plus  $l_t-l_m+1$  fois

## Coût de la recherche simple

En notant  $l_m$  la longueur du motif et  $l_t$  celle de la chaine :

- ullet La boucle for est parcourue au plus  $l_t-l_m+1$  fois
- $\bullet$  Pour chacun de ces parcours, la boucle while interne est parcourue au plus  $l_m$  fois

## Coût de la recherche simple

En notant  $l_m$  la longueur du motif et  $l_t$  celle de la chaine :

- La boucle for est parcourue au plus  $l_t l_m + 1$  fois
- ullet Pour chacun de ces parcours, la boucle while interne est parcourue au plus  $l_m$  fois

Au plus, l'algorithme effectue donc  $l_m(l_t - l_m + 1)$  comparaisons.

### Coût de la recherche simple

En notant  $l_m$  la longueur du motif et  $l_t$  celle de la chaine :

- La boucle for est parcourue au plus  $l_t l_m + 1$  fois
- ullet Pour chacun de ces parcours, la boucle while interne est parcourue au plus  $l_m$  fois

Au plus, l'algorithme effectue donc  $l_m(l_t - l_m + 1)$  comparaisons.

### Exemple

Combien de comparaisons seront nécessaires si on recherche le motif bbbbbbbbb (neuf fois le caractère b suivi d'un a) dans une chaine contenant un million de b?



#### Accélération de la recherche

Supposons qu'on recherche le motif extra dans la chaine un excellent exemple et un exercice extraordinaire. La comparaison naïve ci-dessus commence par :



#### Accélération de la recherche

Supposons qu'on recherche le motif extra dans la chaine un excellent exemple et un exercice extraordinaire. La comparaison naïve ci-dessus commence par :

u	n		е	х	С	е	1	1	е	n	t
<b></b>											
е	х	t.	r	а							



#### Accélération de la recherche

Supposons qu'on recherche le motif extra dans la chaine un excellent exemple et un exercice extraordinaire. La comparaison naïve ci-dessus commence par :

 u
 n
 e
 x
 c
 e
 l
 l
 e
 n
 t

Deux idées vont permettre d'accélérer la recherche :

### Accélération de la recherche

Supposons qu'on recherche le motif extra dans la chaine un excellent exemple et un exercice extraordinaire. La comparaison naïve ci-dessus commence par :

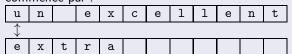


Deux idées vont permettre d'accélérer la recherche :

• Commencer par la fin du motif.

#### Accélération de la recherche

Supposons qu'on recherche le motif extra dans la chaine un excellent exemple et un exercice extraordinaire. La comparaison naïve ci-dessus commence par :



Deux idées vont permettre d'accélérer la recherche :

- Commencer par la fin du motif.
- Prétraiter le motif de façon à éviter des comparaisons inutiles.



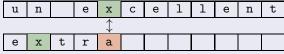
### Accélération de la recherche

Dans l'exemple ci-dessus cela donne :

3. Algorithme de Boyer-Moore

## Accélération de la recherche

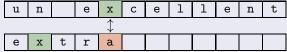
Dans l'exemple ci-dessus cela donne :



3. Algorithme de Boyer-Moore

### Accélération de la recherche

Dans l'exemple ci-dessus cela donne :

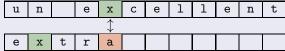


On peut directement décaler le motif de 3 emplacements car le dernier x du motif se trouve à 3 emplacements de la fin du motif.

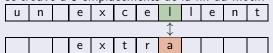
3. Algorithme de Boyer-Moore

### Accélération de la recherche

Dans l'exemple ci-dessus cela donne :



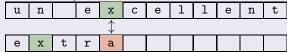
On peut directement décaler le motif de 3 emplacements car le dernier x du motif se trouve à 3 emplacements de la fin du motif.



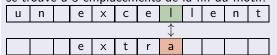
3. Algorithme de Boyer-Moore

#### Accélération de la recherche

Dans l'exemple ci-dessus cela donne :



On peut directement décaler le motif de 3 emplacements car le dernier x du motif se trouve à 3 emplacements de la fin du motif.



Cette fois, le 1 ne se trouve pas dans le motif, on peut donc décaler de la longueur du motif. Et la recherche s'arrête en ayant effectué seulement deux comparaisons.

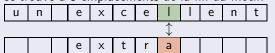


#### Accélération de la recherche

Dans l'exemple ci-dessus cela donne :

	u	n		е	х	С	е	1	1	е	n	t
ľ	\$											
	е	х	t	r	a							

On peut directement décaler le motif de 3 emplacements car le dernier x du motif se trouve à 3 emplacements de la fin du motif.



Cette fois, le 1 ne se trouve pas dans le motif, on peut donc décaler de la longueur du motif. Et la recherche s'arrête en ayant effectué seulement deux comparaisons.

### Visualisation en ligne

Visualisation en ligne du fonctionnement de l'algorithme accéléré

3. Algorithme de Boyer-Moore

## Algorithme de Boyer-Moore-Hoorspool

• La première phase consiste en un prétraitement du motif, afin de construire une fonction de décalage qui indique pour chaque caractère :

3. Algorithme de Boyer-Moore

## Algorithme de Boyer-Moore-Hoorspool

- La première phase consiste en un prétraitement du motif, afin de construire une fonction de décalage qui indique pour chaque caractère :
  - Si le caractère est présent dans le motif alors l'indice de sa dernière occurrence

3. Algorithme de Boyer-Moore

## Algorithme de Boyer-Moore-Hoorspool

- La première phase consiste en un prétraitement du motif, afin de construire une fonction de décalage qui indique pour chaque caractère :
  - Si le caractère est présent dans le motif alors l'indice de sa dernière occurrence
  - Sinon la longueur du motif

3. Algorithme de Boyer-Moore

## Algorithme de Boyer-Moore-Hoorspool

- La première phase consiste en un prétraitement du motif, afin de construire une fonction de décalage qui indique pour chaque caractère :
  - Si le caractère est présent dans le motif alors l'indice de sa dernière occurrence
  - Sinon la longueur du motif
- Ensuite on effectue une recherche en partant de la fin du motif en cas de non correspondance, on décale de la valeur fournie par la fonction de décalage.

### Exemple

Construire la table de décalage du motif "deux"

3. Algorithme de Boyer-Moore

## Algorithme de Boyer-Moore-Hoorspool

- La première phase consiste en un prétraitement du motif, afin de construire une fonction de décalage qui indique pour chaque caractère :
  - Si le caractère est présent dans le motif alors l'indice de sa dernière occurrence
  - Sinon la longueur du motif

### Exemple

- Construire la table de décalage du motif "deux"
- Simuler le fonction de l'algorithme de Boyer-Moore-Hoorspool pour recherche ce motif dans le chaine "