

Approche naïve

Principe  
Implémentation

Approche plus  
efficace :  
Boyer-Moore

Recherche à l'envers  
Décalages par sauts  
Prétraitement du motif  
Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)  
Complexité

# Recherche textuelle

Christophe Viroulaud

Terminale - NSI

**Algo 27**

La recherche textuelle est une fonctionnalité intégrée dans tous les logiciels de traitements de texte.

#### Approche naïve

Principe

Implémentation

#### Approche plus efficace : Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

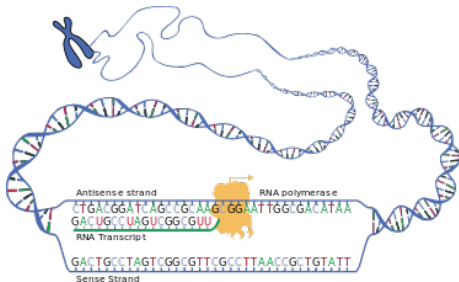


FIGURE 1 – Des applications multiples

- 4 bases nucléiques : Adénine, Cytosine, Guanine, Thymine,
- ADN humain : 3 milliards de bases répartis sur 23 paires de chromosomes.

## Remarque

À titre de comparaison un roman compte environ 500000 caractères.

### Approche naïve

Principe

Implémentation

### Approche plus efficace :

Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

## Approche naïve

Principe

Implémentation

## Approche plus efficace : Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

Comment effectuer une recherche textuelle efficace ?

## 1. Approche naïve

### 1.1 Principe

### 1.2 Implémentation

## 2. Approche plus efficace : Boyer-Moore

### Approche naïve

Principe

Implémentation

### Approche plus efficace : Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

# Approche naïve - Principe

- ▶ observer une **fenêtre** du texte,
- ▶ dans cette fenêtre, comparer chaque lettre du **motif** recherché au texte,
- ▶ décaler la fenêtre d'un cran dès qu'il n'y a pas de correspondance.

	0	1	2	3	4	5	6	7	8	9	10
texte	a	c	g	a	t	c	c	a	t	g	a
motif	c	a	t								
	0	1	2								

## Approche naïve

### Principe

### Implémentation

## Approche plus

### efficace :

### Boyer-Moore

### Recherche à l'envers

### Décalages par sauts

### Prétraitement du motif

### Algorithme de Boyer-Moore (simplifié - version Horspool)

### Complexité

## Approche naïve

## Principe

## Implémentation

Approche plus  
efficace :

## Boyer-Moore

## Recherche à l'envers

## Décalages par sauts

## Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

## Complexité

	0	1	2	3	4	5	6	7	8	9	10
texte	a	c	g	a	t	c	c	a	t	g	a
motif	c	a	t								
	0	1	2								

FIGURE 2 – Première comparaison : pas de correspondance

Décalage de la fenêtre

## Approche naïve

### Principe

Implémentation

## Approche plus efficace : Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

	0	1	2	3	4	5	6	7	8	9	10
texte	a	c	g	a	t	c	c	a	t	g	a
motif		c	a	t							
		0	1	2							

FIGURE 3 – Première comparaison : correspondance



## Approche naïve

Principe

Implémentation

Approche plus  
efficace :

Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

	0	1	2	3	4	5	6	7	8	9	10
texte	a	c	g	a	t	c	c	a	t	g	a
motif		c	a	t							
		0	1	2							

FIGURE 4 – Deuxième comparaison : pas de correspondance

Décalage de la fenêtre

## 1. Approche naïve

### 1.1 Principe

### 1.2 Implémentation

## 2. Approche plus efficace : Boyer-Moore

### Approche naïve

Principe

**Implémentation**

### Approche plus efficace : Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

## Approche naïve

Principe

Implémentation

## Approche plus

efficace :

Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

### Activité 1 :

1. Écrire la fonction `recherche_naive(texte: str, motif: str) → int` qui renvoie la position du *motif* dans le *texte* ou -1 s'il n'est pas présent.
2. Estimer la complexité temporelle de cet algorithme dans le pire des cas : le motif n'est pas présent dans le texte.

## Approche naïve

Principe

Implémentation

## Approche plus

efficace :

Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

```
1 def recherche_naive(texte: str, motif: str) -> int:
2     """
3     renvoie la position du motif dans le texte
4     -1 s'il n'est pas présent
5     """
6     # dernière position = taille(texte) - taille(
motif)
7     for i in range(len(texte)-len(motif)+1):
8         j = 0
9         while j < len(motif) and
0             motif[j] == texte[i+j]:
1             j += 1
2         # correspondance sur toute la fenêtre
3         if j == len(motif):
4             return i
5     return -1
```

Imaginons le cas :

	0	1	2	3	4	5	6	7	8	9	10
texte	a	a	a	a	a	a	a	a	a	a	t
motif	a	a	t								
	0	1	2								

- On vérifie toute la fenêtre à chaque fois.
- À chaque **non correspondance** la fenêtre avance de 1.
- La complexité dépend de la taille du texte et de celle du motif.

## Approche naïve

Principe

Implémentation

## Approche plus efficace :

### Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

## 1. Approche naïve

## 2. Approche plus efficace : Boyer-Moore

### 2.1 Recherche à l'envers

### 2.2 Décalages par sauts

### 2.3 Prétraitement du motif

### 2.4 Algorithme de Boyer-Moore (simplifié - version Horspool)

### 2.5 Complexité

#### Approche naïve

Principe

Implémentation

#### Approche plus efficace : Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

## Approche naïve

Principe

Implémentation

## Approche plus efficace : Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

- 1970 : algorithme de Knuth-Morris-Pratt.  $O(T + M)$

## Approche naïve

Principe

Implémentation

## Approche plus efficace : Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

- ▶ 1970 : algorithme de Knuth-Morris-Pratt.  $O(T + M)$
- ▶ 1977 : algorithme de Boyer-Moore.
  - ▶ meilleur des cas :  $O(T/M)$
  - ▶ pire des cas :  $O(T + M)$



## Approche naïve

Principe

Implémentation

## Approche plus efficace : Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

- ▶ 1970 : algorithme de Knuth-Morris-Pratt.  $O(T + M)$
- ▶ 1977 : algorithme de Boyer-Moore.
  - ▶ meilleur des cas :  $O(T/M)$
  - ▶ pire des cas :  $O(T + M)$
- ▶ 1980 : Horspool propose une version simplifiée de l'algorithme de Boyer-Moore.  $O(T)$

## Approche naïve

Principe

Implémentation

## Approche plus efficace :

Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

La première idée de cet algorithme est de commencer la recherche **en partant de la fin du motif**.

	0	1	2	3	4	5	6	7	8	9	10
texte	a	c	g	a	t	c	c	a	t	g	a
motif	c	a	t								
	0	1	2								

FIGURE 5 – Première comparaison : pas de correspondance

Approche naïve

Principe  
Implémentation

Approche plus  
efficace :  
Boyer-Moore

Recherche à l'envers  
Décalages par sauts  
Prétraitement du motif  
Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)  
Complexité

## Remarque

Pour l'instant cette approche ne semble pas apporter d'amélioration par rapport à l'algorithme précédent.

## 1. Approche naïve

## 2. Approche plus efficace : Boyer-Moore

### 2.1 Recherche à l'envers

### 2.2 Décalages par sauts

### 2.3 Prétraitement du motif

### 2.4 Algorithme de Boyer-Moore (simplifié - version Horspool)

### 2.5 Complexité

#### Approche naïve

Principe

Implémentation

#### Approche plus efficace : Boyer-Moore

Recherche à l'envers

#### Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

## Approche naïve

Principe

Implémentation

## Approche plus efficace :

Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

Le motif ne contient pas la lettre **g** (la dernière lettre de la fenêtre).

	0	1	2	3	4	5	6	7	8	9	10
texte	a	c	g	a	t	c	c	a	t	g	a
motif	c	a	t								
	0	1	2								

FIGURE 6 – Comparaisons inutiles

## Approche naïve

Principe

Implémentation

## Approche plus efficace : Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

	0	1	2	3	4	5	6	7	8	9	10
texte	a	c	g	a	t	c	c	a	t	g	a
motif		c	a	t							
		0	1	2							

FIGURE 7 – Comparaison inutile

## Approche naïve

Principe

Implémentation

Approche plus  
efficace :

Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

	0	1	2	3	4	5	6	7	8	9	10
texte	a	c	g	a	t	c	c	a	t	g	a
motif			c	a	t						
			0	1	2						

FIGURE 8 – Comparaison inutile

## Approche naïve

Principe

Implémentation

## Approche plus

efficace :

Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

On peut donc directement décaler le motif à l'indice 3 du texte.

	0	1	2	3	4	5	6	7	8	9	10
texte	a	c	g	a	t	c	c	a	t	g	a
motif				c	a	t					
				0	1	2					

FIGURE 9 – Décalage par saut



On n'observe pas de correspondance par contre la lettre **c** existe dans le motif. On va donc le décaler pour les faire coïncider.

	0	1	2	3	4	5	6	7	8	9	10
texte	a	c	g	a	t	c	c	a	t	g	a
motif				c	a	t					
				0	1	2					

FIGURE 10 – Nouvelle situation

#### Approche naïve

Principe

Implémentation

#### Approche plus efficace :

##### Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

## Approche naïve

Principe

Implémentation

## Approche plus efficace :

### Boyer-Moore

Recherche à l'envers

#### Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

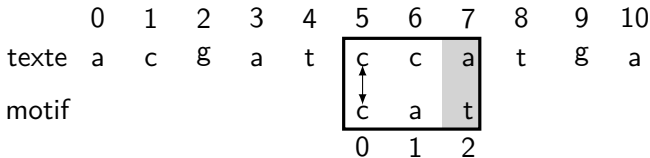


FIGURE 11 – Décalage par saut

## À retenir

On décale la position de recherche dans le texte en fonction de la dernière lettre de la fenêtre.

### Approche naïve

Principe  
Implémentation

### Approche plus efficace : Boyer-Moore

Recherche à l'envers  
Décalages par sauts  
Prétraitement du motif  
Algorithme de Boyer-Moore  
(simplifié - version Horspool)  
Complexité

## 1. Approche naïve

## 2. Approche plus efficace : Boyer-Moore

### 2.1 Recherche à l'envers

### 2.2 Décalages par sauts

### 2.3 Prétraitement du motif

### 2.4 Algorithme de Boyer-Moore (simplifié - version Horspool)

### 2.5 Complexité

#### Approche naïve

Principe

Implémentation

#### Approche plus efficace : Boyer-Moore

Recherche à l'envers

Décalages par sauts

**Prétraitement du motif**

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

## À retenir

Pour pouvoir décaler par saut, il faut connaître la dernière position de chaque lettre dans le motif. Le prétraitement consiste à calculer le décalage à appliquer pour amener chaque caractère du motif à la place du dernier caractère.

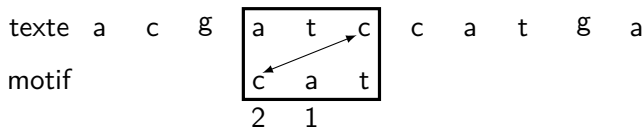


FIGURE 12 – Calculs des décalages

### Approche naïve

Principe

Implémentation

### Approche plus efficace :

#### Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

## Remarque

On ne regarde pas la dernière position de la clé (la lettre *t* ici). Sinon la distance associée serait nulle et on resterait sur place après l'avoir lue dans le texte.

texte	a	c	g	a	t	c	c	a	t	g	a
motif			c	a	t						
			2	1	0						

FIGURE 13 – Sauf la dernière lettre

### Approche naïve

Principe

Implémentation

### Approche plus efficace :

#### Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

## À retenir

Dans le cas de la répétition d'un caractère, on garde la distance la plus courte.

texte	a	c	g	a	t	c	c	a	t	g	a
motif	t	a	c	a	t						
	4		2	1							

FIGURE 14 – Répétition dans le motif

### Approche naïve

Principe

Implémentation

### Approche plus efficace :

#### Boyer-Moore

Recherche à l'envers

Décalages par sauts

#### Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

## Approche naïve

Principe

Implémentation

## Approche plus efficace :

### Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

**Activité 2 :** Écrire la fonction  
`pretraitement_decalages(motif: str) → dict`  
qui associe chaque lettre du motif (sauf la dernière) à  
son décalage.



## Approche naïve

Principe

Implémentation

## Approche plus

efficace :

Boyer-Moore

Recherche à l'envers

Décalages par sauts

**Prétraitement du motif**

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

```
1 def pretraitement_decalages(motif: str) -> dict:
2     """
3     renvoie le dictionnaire des décalages à
4     appliquer
5     pour chaque lettre du motif (sauf dernière)
6     """
7     decalages = dict()
8     # on s'arrête à l'avant dernière lettre du motif
9     for i in range(len(motif)-1):
10         # la distance est mise à jour en cas de répé
11         tition
12         decalages[motif[i]] = len(motif)-1-i
13     return decalages
```

## 1. Approche naïve

## 2. Approche plus efficace : Boyer-Moore

### 2.1 Recherche à l'envers

### 2.2 Décalages par sauts

### 2.3 Prétraitement du motif

### 2.4 Algorithme de Boyer-Moore (simplifié - version Horspool)

### 2.5 Complexité

#### Approche naïve

Principe

Implémentation

#### Approche plus efficace : Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

**Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)**

Complexité

L'algorithme de Boyer-Moore s'écrit alors :

```
1 Créer le tableau des décalages
2 Tant qu'on n'est pas à la fin du texte
3     Comparer le motif à la position du texte
4     Si le motif est présent
5         Renvoyer la position
6     Sinon
7         Décaler la fenêtre
8 Renvoyer -1 si le motif n'est pas présent
```

Code 1 – Algorithme de Boyer-Moore (version Horspool)

Approche naïve

Principe

Implémentation

Approche plus  
efficace :

Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

### Activité 3 :

1. Écrire la fonction `compare(texte: str, position: int, motif: str) → bool` qui renvoie *True* si le motif est présent à la position *i* du texte.
2. Écrire la fonction `decalage_fenetre(decalages: dict, taille: int, lettre: str) → int` qui renvoie le décalage à appliquer pour faire coïncider le motif à la dernière *lettre* de la fenêtre. Si la lettre n'est pas présente, la *taille* du motif est renvoyée.
3. Écrire alors la fonction `boyer_moore(texte: str, motif: str) → int` qui renvoie la position du motif dans le texte et -1 sinon.

#### Approche naïve

Principe

Implémentation

#### Approche plus efficace :

##### Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

```
1 def compare(texte: str, position: int, motif: str) -> bool:
2     """
3     compare le morceau du texte
4     (en partant de position + taille(motif))
5     avec le motif
6
7     Returns:
8         bool: True si on a trouvé le motif
9     """
10    # position de la dernière lettre de la fenêtre
11    en_cours = position + len(motif) - 1
12    # parcours de la fenêtre à l'envers
13    for i in range(len(motif) - 1, -1, -1):
14        if not(texte[en_cours] == motif[i]):
15            return False
16        else:
17            en_cours -= 1
18    return True
```

Approche naïve

Principe

Implémentation

Approche plus

efficace :

Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

## Approche naïve

Principe

Implémentation

## Approche plus

efficace :

Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

```
def decalage_fenetre(decalages: dict, taille: int, lettre: str) ->
    int:
    """
    renvoie la valeur du décalage à appliquer.
    si la lettre n'est pas dans le tableau
    c'est la taille du motif qui est appliqué

    Args:
        decalages (dict): dico des décalages
        taille (int): taille du motif (= décalage max)
        lettre (str): dernière lettre de la fenêtre

    Returns:
        int: décalage à appliquer
    """
    for cle, val in decalages.items():
        if cle == lettre:
            return val
    # si la lettre n'est pas dans le dico (= le motif)
    return taille
```

## Approche naïve

Principe

Implémentation

## Approche plus

efficace :

Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

```
1 def decalage_fenetre2(decalages: dict, taille: int, lettre: str) ->  
  int:  
2     # la méthode get renvoie une valeur par défaut si elle ne  
  trouve pas la clé  
3     return decalages.get(lettre, taille)  
4  
5  
6 def decalage_fenetre3(decalages: dict, taille: int, lettre: str) ->  
  int:  
7     try:  
8         res = decalages[lettre]  
9     except KeyError:  
10        res = taille  
11    return res
```

## Approche naïve

Principe

Implémentation

Approche plus  
efficace :  
Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

```
1 def boyer_moore(texte: str, motif: str) -> int:
2     """
3     Returns:
4         int: la position du motif dans le texte, -1 sinon.
5     """
6     decalages = pretraitement_decalages(motif)
7     i = 0
8     while i <= len(texte)-len(motif):
9         # si on trouve le motif
10        if compare(texte, i, motif):
11            return i
12        else:
13            # sinon on décale (en fonction de la dernière
14            # lettre de la fenêtre)
15            decale = decalage_fenetre(decalages,
16                                     len(motif),
17                                     texte[i+len(motif)-1])
18            i += decale
19        # si on sort de la boucle, on n'a rien trouvé
20    return -1
```



## 1. Approche naïve

## 2. Approche plus efficace : Boyer-Moore

### 2.1 Recherche à l'envers

### 2.2 Décalages par sauts

### 2.3 Prétraitement du motif

### 2.4 Algorithme de Boyer-Moore (simplifié - version Horspool)

### 2.5 Complexité

#### Approche naïve

Principe

Implémentation

#### Approche plus efficace : Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

Intuitivement l'algorithme semble plus rapide que la version naïve car il ne teste pas toutes les lettres du texte.

a	a	a	a	b	a	a	a	a	b	a	a	a	a	b
c	c	c	c	c										

FIGURE 15 – Un cas représentatif

## Approche naïve

Principe

Implémentation

## Approche plus efficace :

### Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

## Approche naïve

Principe

Implémentation

## Approche plus efficace : Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

a a a a b a a a a b a a a a b  
c c c c c

a a a a b a a a a b a a a a b  
c c c c c

FIGURE 16 – Algorithme naïf

## Approche naïve

Principe

Implémentation

## Approche plus efficace : Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

a a a a b a a a a b a a a a b  
c c c c c

a a a a b a a a a b a a a a b  
c c c c c

FIGURE 17 – Algorithme de Boyer-Moore

Approche naïve

Principe

Implémentation

Approche plus  
efficace :  
Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

**Activité 4** : Compter le nombre d'itérations de la recherche avec l'algorithme naïf puis celui de Boyer-Moore.

## Approche naïve

Principe

Implémentation

## Approche plus efficace : Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

- ▶ Algorithme naïf : 10 décalages,
- ▶ Algorithme de Boyer-Moore : 3 décalages.

Approche naïve

Principe

Implémentation

Approche plus  
efficace :

Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

## Remarques

- ▶ Dans le meilleur des cas, la complexité temporelle de l'algorithme est  $O(T/M)$  où  $T$  est la taille du texte et  $M$  celle du motif.
- ▶ Plus le motif est long plus l'algorithme est rapide.

Le prétraitement a un coût (temporel et spatial) mais qui est grandement compensé.

#### Approche naïve

Principe

Implémentation

#### Approche plus efficace : Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité



# Cas critique

## Approche naïve

Principe

Implémentation

## Approche plus efficace : Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

texte	t	a	a	a	a	a	a	a	a	a	a
motif	a	a	a	a	a	a	a				

texte	t	a	a	a	a	a	a	a	a	a	a
motif		a	a	a	a	a	a	a			

FIGURE 18 – Cas critique

## Approche naïve

Principe

Implémentation

## Approche plus efficace : Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore  
(simplifié - version  
Horspool)

Complexité

Les programmes du cours sont téléchargeables [ici](#).