Recherche textuelle

Recherche textuelle

Christophe Viroulaud

Algo 27

Recherche textuelle

Christophe Viroulaud

Terminale - NSI

Algo 27

Recherche textuelle

Principe Implémentation

> cace : yer-Moore

> > ages par sauts aitement du motif ithme de Boyer-Moore

Recherche textuelle

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

Recherche textuelle





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.

Recherche textuelle

Principe

Approche plus fficace : Boyer-Moore

Décalages par sauts Prétraitement du motif Algorithme de Boyer-Moore (simplifié - version Recherche textuelle

Comment effectuer une recherche textuelle efficace?

Comment effectuer une recherche textuelle efficace?

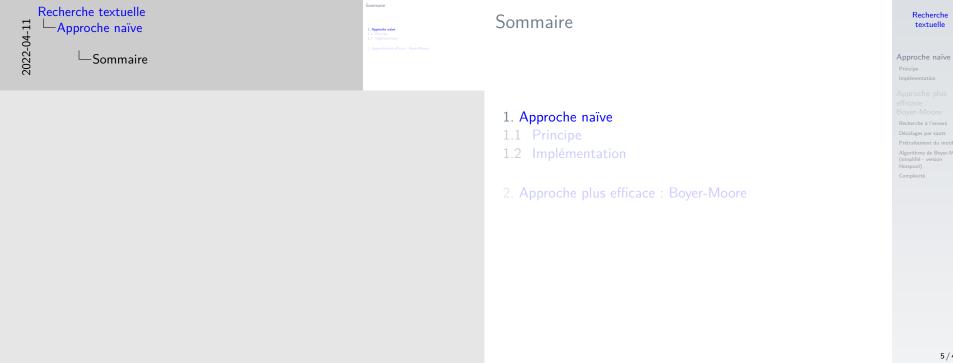
Recherche textuelle

Principe Implémentation

> cace : yer-Moore

echerche à l'envers écalages par sauts

aitement du motif



5 / 45

Approche naïve - Principe

- - ► observer une fenêtre du texte recherché au texte,
 - ► décaler la fenêtre d'un cran dès qu'il n'v a pas de
 - texte a c 8 a t c c a t 8 a

Approche naïve - Principe

- b 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.

Recherche textuelle

Principe



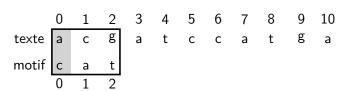


FIGURE 2 – Première comparaison : pas de correspondance

Décalage de la fenêtre

Recherche textuelle

Approche naïve

Principe Implémentation

Approche plus

etticace : Boyer-Moore

Décalages par sauts Prétraitement du motif Algorithme de Boyer-Moc

rithme de Boyer-Moore plifié - version pool) plexité

FIGURE 3 – Première comparaison : correspondance

Recherche textuelle

Approche naïve
Principe

Implémentation

pproche plus ficace : oyer-Moore

Recherche a l'envers Décalages par sauts Prétraitement du motif Algorithme de Boyer-Moo (simplifié - version

FIGURE 4 – Deuxième comparaison : pas de correspondance

Décalage de la fenêtre

Recherche textuelle

Approche naïve

Principe Implémentation

Approche plus
efficace :
Boyer-Moore

Recherche à l'envers Décalages par sauts Prétraitement du motif

traitement du motif orithme de Boyer-Moo nplifié - version spool)



Sommaire

- 1. Approche naïve
- 1.2 Implémentation
- 2 Approche plus efficace : Bover-Moore

Recherche

textuelle

Implémentation

Implémentation

Activité 1

. Écrire la fonction recherche maive(texte: str, motif: str) -> int qui renvoie la

position du motif dans le texte ou -1 s'il n'est pas Estimer la complexité temporelle de cet algorithme

dans le pire des cas : le motif n'est pas présent

Activité 1:

- 1. Écrire la fonction recherche naive(texte: str, motif: str) \rightarrow 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.

Correction

```
def recherche naive(texte: str, motif: str) -> int:
    renvoie la position du motif dans le texte
    -1 s'il n'est pas présent
    # dernière position = taille(texte) - taille(
   motif)
    for i in range(len(texte)-len(motif)+1):
        j = 0
        while j < len(motif) and</pre>
                motif[j] == texte[i+j]:
            j += 1
        # correspondance sur toute la fenêtre
        if j == len(motif):
            return i
    return -1
```

Recherche textuelle

Approche naive

Implémentation

proche plus

rer-Moore nerche à l'envers

> es par sauts ement du motif me de Boyer-Moore é - version

Correction

Imaginons le cas :

- ► 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.

proche naïve

Recherche

textuelle

Implémentation

fficace :
Soyer-Moore

Recherche à l'envers Décalages par sauts

étraitement du motif gorithme de Boyer-Moore implifié - version



14 / 45

Bover-Moore

- └─Boyer-Moore
- 1. évoquera la complexité de Boyer-Moore en fin de cours
- 2. version Horspool est simplifiée mais pas forcément aussi efficace que BM

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

oche naïve

Recherche

textuelle

nplémentation

Approche plus efficace : Boyer-Moore

lages par sauts
raitement du motif
rithme de Boyer-Moore

Recherche textuelle
Approche plus efficace : Boyer-Moore

-Boyer-Moore

- 1. évoquera la complexité de Boyer-Moore en fin de cours
- 2. version Horspool est simplifiée mais pas forcément aussi efficace que BM

Bover-Moore

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

Boyer-Moore

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

► 1977 : algorithme de Boyer-Moore.

ightharpoonup meilleur des cas : O(T/M)

ightharpoonup pire des cas : O(T+M)

Recherche textuelle

rincipe nplémentation

Approche plus efficace :
Boyer-Moore

Décalages par sauts
Prétraitement du motif
Algorithme de Boyer-Moore

ifié - version ool) lexité Recherche textuelle -Approche plus efficace : Boyer-Moore

-Boyer-Moore

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

1980 : Horspool propose une version simplifiée de l'algorithme de Bover-Moore, O(T)

Bover-Moore

- 1. évoquera la complexité de Boyer-Moore en fin de cours
- 2. version Horspool est simplifiée mais pas forcément aussi efficace que BM

Boyer-Moore

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



Recherche à l'envers

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	а	С	g	а	t	С	С	а	t	g	a	
motif	С	а	t									
	0	1	2									

FIGURE 5 – Première comparaison : pas de correspondance

Recherche textuelle

Principe

Approcne plus efficace : Bover-Moore

Recherche à l'envers

Décalages par caute

Prétraitement du motif

orithme de Boyer-Moore oplifié - version opool) oplexité

Remarque

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

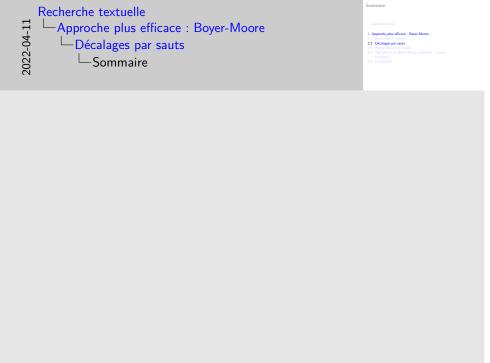
Recherche textuelle

incipe plémentation

Approche plus efficace : Boyer-Moore

Recherche à l'envers

étraitement du motif gorithme de Boyer-Moore



Sommaire

- 2. Approche plus efficace : Boyer-Moore
- 2.1 Recherche à l'envers
- 2.2 Décalages par sauts

- 2.4 Algorithme de Boyer-Moore (simplifié version

18 / 45

Recherche

textuelle

Décalages par sauts

Décalage par sauts

Décalage par sauts

Le motif ne contient pas la lettre \mathbf{g} (la dernière lettre de la fenêtre).

 $\label{eq:Figure 6-Comparaisons in utiles} Figure 6-Comparaisons in utiles$

Recherche textuelle

Principe mplémentation

Approcne plus efficace : Bover-Moore

Recherche à l'envers Décalages par sauts

rétraitement du motif

orithme de Boyer-Moore oplifié - version spool) oplexité



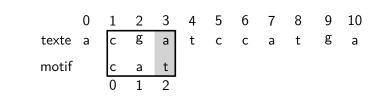


FIGURE 7 – Comparaison inutile

Recherche

textuelle

Décalages par sauts



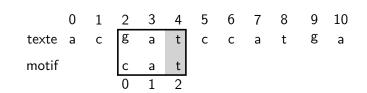


FIGURE 8 – Comparaison inutile

textuelle

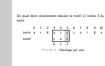
Recherche

Principe Principe

Approche plus efficace :
Boyer-Moore

Recherche à l'envers Décalages par sauts

Prétraitement du motif Algorithme de Boyer-Moor (simplifié - version



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

 ${\rm Figure} \ 9 - {\rm D\acute{e}calage} \ {\rm par} \ {\rm saut}$

Recherche textuelle

Approcne naive

nplémentation

efficace : Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif Algorithme de Boyer-Moor



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

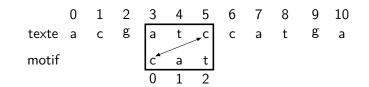


FIGURE 10 – Nouvelle situation

Recherche textuelle

Approcne naive

mplémentation

Approche plus efficace :
Boyer-Moore

Recherche à l'envers

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moore

(simplifié - version

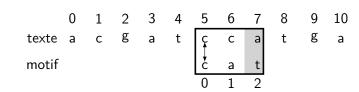


FIGURE 11 – Décalage par saut

Recherche textuelle

Approche naive

Implémentation

efficace : Boyer-Moore

Décalages par sauts

Prétraitement du motif Algorithme de Boyer-Moor (simplifié - version



- 1. et de sa présence éventuelle dans le motif
- 2. il existe variante *bad char* : on décale en fonction du caractère qui ne correspond pas

À retenir

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

Approcne naive

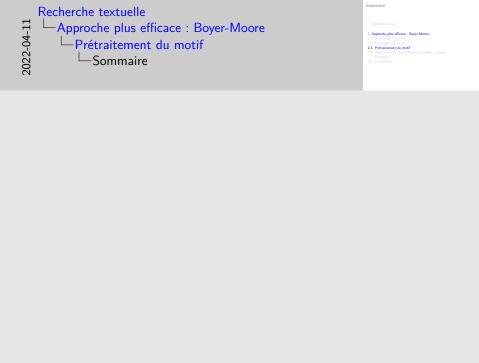
Implémentation

efficace : Boyer-Moore

cherche à l'envers

Décalages par sauts

gorithme de Boyer-Moore mplifié - version rspool)



Sommaire

- 1. Approche naïve
- 2. Approche plus efficace : Boyer-Moore
- 2.1 Recherche à l'envers
- 2.2 Décalages nar sauts
- 2.3 Prétraitement du motif
- 2.3 Pretraitement du mot
- 2.4 Algorithme de Boyer-Moore (simplifié version
- Horspool)

Recherche

textuelle

Prétraitement du motif

Recherche textuelle

Approche plus efficace : Boyer-Moore

Prétraitement du motif

Prétraitement du motif



Prétraitement du motif

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.

texte a c g a t c c a t g a motif

FIGURE 12 – Calculs des décalages

Recherche textuelle

Approche naive

Implémentation

pproche plus fficace : oyer-Moore

Recherche à l'envers

Prétraitement du motif

Algorithme de Bover-Moor

implifié - version orspool) omplexité

Recherche textuelle Approche plus efficace : Boyer-Moore Prétraitement du motif



attention t pourrait être présent ailleurs dans motif \rightarrow on prend en compte alors

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.

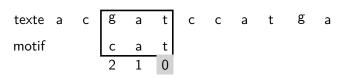


FIGURE 13 – Sauf la dernière lettre

Recherche textuelle

Approche naïve

Implémentation

pproche plus ficace : oyer-Moore

Recherche à l'envers

Prétraitement du motif

Algorithme de Boyer-Moore

nplifié - version spool) nplexité

Recherche textuelle Approche plus efficace : Boyer-Moore Prétraitement du motif



on fait coı̈ncider le premier t du motif avec la dernière lettre de la fenêtre

À retenir

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

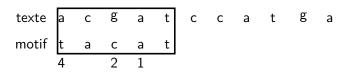


FIGURE 14 – Répétition dans le motif

Recherche textuelle

Approche naïve

nplémentation

Approche plus fficace :

Recherche à l'envers

Prétraitement du motif

Algorithme de Boyer-Moor (simplifié - version Activité 2 : Écrire la fonction pretraitement_decalages(motif: str) \rightarrow dict qui associe chaque lettre du motif (sauf la dernière) à son décalage.

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

Recherche textuelle

Principe Principe

nplémentation

ficace : oyer-Moore

Décalages par sauts

Prétraitement du motif Algorithme de Boyer-Moore (simplifié - version

Correction

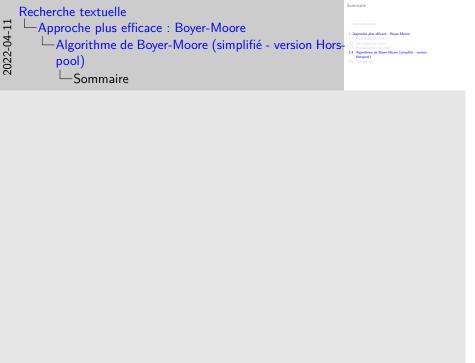
def pretraitement decalages(motif: str) -> dict:

```
Prétraitement du motif
```

Recherche

textuelle

```
renvoie le dictionnaire des décalages à
appliquer
 pour chaque lettre du motif (sauf dernière)
 11 11 11
decalages = dict()
 # on s'arrête à l'avant dernière lettre du motif
 for i in range(len(motif)-1):
     # la distance est mise à jour en cas de répé
tition
     decalages[motif[i]] = len(motif)-1-i
 return decalages
```



Sommaire

- 2. Approche plus efficace : Boyer-Moore
- 2.1 Recherche à l'envers

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

Recherche

textuelle

Algorithme de Boyer-Moore (simplifié - version Horspool)

```
Recherche textuelle

Approche plus efficace : Boyer-Moore

Algorithme de Boyer-Moore (simplifié - version Horspool)

Algorithme de Boyer-Moore
```

Algorithme de Boyen-Moore 'sirit alon :

L'agenthme de Boyen-Moore (activate de Boyen-Moore (activ

Horspool : en 1980, version simplifiée de Boyer-Moore ; il existe plusieurs versions.

Algorithme de Boyer-Moore

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

```
Créer le tableau des décalages

Tant qu'on n'est pas à la fin du texte

Comparer le motif à la position du texte

Si le motif est présent

Renvoyer la position

Sinon

Décaler la fenêtre

Renvoyer -1 si le motif n'est pas présent
```

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

Recherche textuelle

Principe

Approche plu

Boyer-Moore
Recherche à l'envers

Prétraitement du motif

Algorithme de Boyer-Moore

(simplifié - version Horspool) Complexité

Recherche textuelle Approche plus efficace : Boyer-Moore Algorithme de Boyer-Moore (simplifié - version Horspool)

themed 1:

Ecrim is forction compare(texte: str., persistant int, metf: str) -> bad qui prestitant int, metf: str) -> bad qui me de mende expenset à product de toute.

Frierie li forction decalege, centre(decaleger: dite; Frierie li forction decalege, centre(decaleger: dite; met de la decalege; centre(decaleger: dite; met de la decaleger: decentre decaleger: decentre decaleger: decentre decaleger: decale

position du motif dans le texte et -1 sinon.

Activité 3:

- 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.
- 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.

Recherche textuelle

Approche naïve Principe

pproche plus

Recherche à l'envers Décalages par sauts

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

Correction

```
def compare(texte: str, position: int, motif: str)
   -> bool:
    # position de la dernière lettre de la fenêtre
    en_cours = position+len(motif)-1
    # parcours de la fenêtre à l'envers
    for i in range(len(motif)-1, -1, -1):
        if not(texte[en_cours] == motif[i]):
            return False
        else:
            en cours -= 1
    return True
```

Recherche textuelle

rincipe nplémentation

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

```
def decalage_fenetre(decalages: dict, taille: int,
    lettre: str) -> int:
    for cle, val in decalages.items():
        if cle == lettre:
            return val
        # si la lettre n'est pas dans le dico (= le motif)
    return taille
```

ncipe lémentation

coper-Moore

Prétraitement du motif Algorithme de Boyer-Moore (simplifié - version Horspool)

```
Recherche textuelle

Approche plus efficace : Boyer-Moore

Algorithme de Boyer-Moore (simplifié - version Hors-
pool)
```

```
def decalage_fenetre2(decalages: dict, taille: int,
    lettre: str) -> int:
    # la méthode get renvoie une valeur par défaut
    si elle ne trouve pas la clé
    return decalages.get(lettre, taille)
```

```
def decalage_fenetre3(decalages: dict, taille: int,
    lettre: str) -> int:
    try:
        res = decalages[lettre]
    except KeyError:
        res = taille
    return res
```

Code 2 – Variantes

textuelle

Recherche

Principe
Implémentation

ficace : oyer-Moore echerche à l'envers

Décalages par sauts
Prétraitement du motif
Algorithme de Boyer-Moore

(simplifié - version Horspool) Complexité

```
Recherche textuelle

Approche plus efficace : Boyer-Moore

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

```
Out Department out, mail: sty > inc.

1 * Que * present (mail: sty > inc.

1 * Que * present (mail: mail: mail: mail: mail: mail: sty > inc.

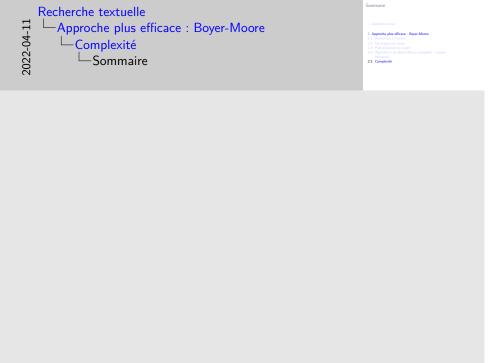
1 * Que * present (mail: mail: m
```

```
def boyer moore(texte: str, motif: str) -> int:
    decalages = pretraitement decalages(motif)
    i = 0
    while i <= len(texte)-len(motif):</pre>
        # si on trouve le motif
        if compare(texte, i, motif):
            return i
        else:
            # sinon on décale (en fonction de la
   dernière lettre de la fenêtre)
            decale = decalage_fenetre(
                             decalages,
                             len(motif),
                             texte[i+len(motif)-1]
            i += decale
    # si on sort de la boucle, on n'a rien trouvé
    return -1
```

Principe Implémentation

> cace : /er-Moore herche à l'envers

Algorithme de Boyer-Moore (simplifié - version Horspool)



Sommaire

- . Approche naïve
- 2. Approche plus efficace : Boyer-Moore
- 2.1 Recherche à l'envers
- 2.2 Décalages par sauts
- 2.3 Prétraitement du moti
- 2.4 Algorithme de Boyer-Moore (simplifié version
 - Horspool)
- 2.5 Complexité

39 / 45

Recherche

textuelle

2022-04-11

Complexité

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

aaabaaabaaab cccc

FIGURE 15 – Un cas représentatif

Recherche textuelle

Principe

mplémentation

ficace :
oyer-Moore
echerche à l'envers

Décalages par sauts

Complexité

ithme de Boyer-Moore difié - version pool)

- 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 b a a a b b c c c c c

FIGURE 16 – Algorithme naïf

Observation

L'algorithme naïf effectue 10 décalages.

Recherche

textuelle

Recherche textuelle
Approche plus efficace : Boyer-Moore
Complexité



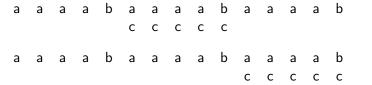


FIGURE 17 – Algorithme de Boyer-Moore

Observation

L'algorithme de Boyer-Moore-Horspool effectue 3 décalages.

Recherche

textuelle

Recherche textuelle

Approche plus efficace : Boyer-Moore

Complexité

Remarques

- ▶ Dans le meilleur des cas, la complexité temporelle de l'algorithme est O(T/M) où T est la taille du tente et M celle du metil
- 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.

- 1. complexité sous-linéaire!
- 2. Naïf O(T.M)
- 3. Complexité moyenne : O(3.M) démontrée par Richard Cole en 1991.
- 4. coût spatial si alphabet grand.

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.

Recherche textuelle

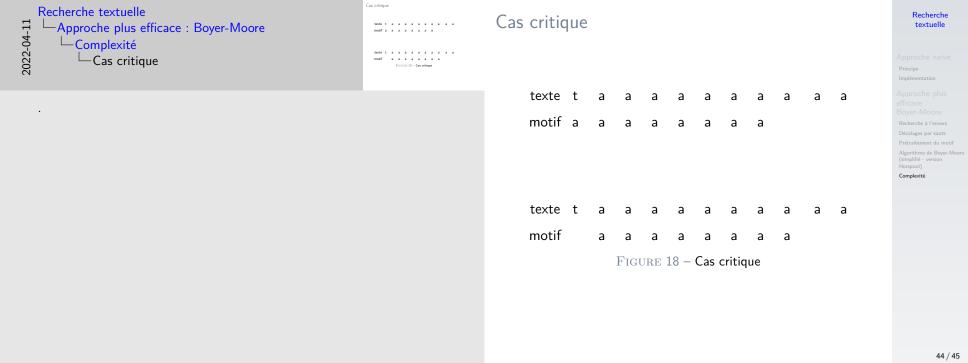
Principe Principe

plémentation

efficace : Boyer-Moore

Décalages par sauts

Prétraitement du motif Algorithme de Boyer-Moore simplifié - version



Remarque

En plus d'un traitement du mauvais caractère, la version complète de l'algorithme de Boyer-Mocre recherche des bons suffixes dans le motif.

Remarque

En plus d'un traitement du mauvais caractère, la version complète de l'algorithme de Boyer-Moore recherche des bons suffixes dans le motif.

Recherche textuelle

Approcne naive

Implémentation

Approche plus efficace : Boyer-Moore

écalages par sauts

retraitement du motif Igorithme de Boyer-Moore implifié - version