Recherche textuelle

Recherche textuelle

Christophe Viroulaud

Terreirale - NSI

Recherche textuelle

Christophe Viroulaud

Terminale - NSI

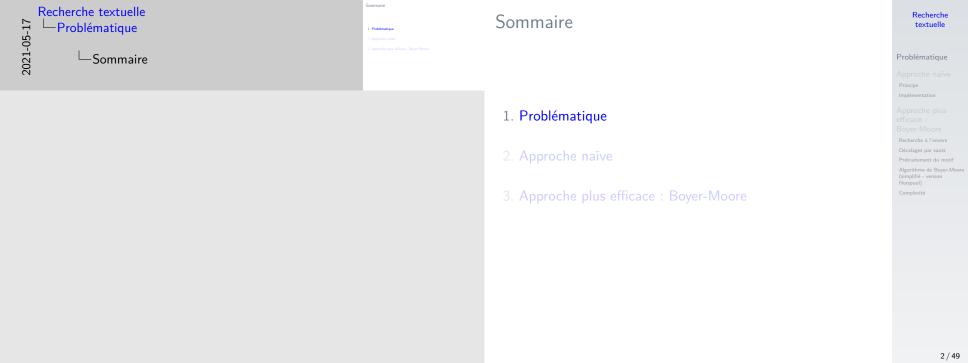
plus

proche plus cace : yer-Moore

Recherche

textuelle

ages par sauts aitement du motif ithme de Boyer-Moore lifié - version





-Des applications multiples

► ADN humain : 3 milliards de bases répartis sur 23 paires

Des applications multiples

roman = 500 000 caractères

Des applications multiples

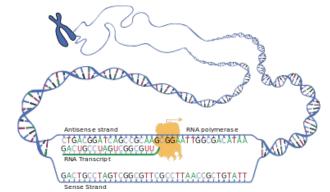


FIGURE 1 – Séquence ADN

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

Problématique

Recherche

textuelle

Recherche

textuelle

Problématique

plexité



Principe

recherché au texte,

indicaler la fenêtre d'un cran dès qu'il n'y a pas de

- 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.

				3							
texte motif	а	С	g	а	t	С	С	а	t	g	а
motif	С	a	t								

Recherche textuelle

i iobiematique

Approche naiv
Principe

mplémentation

pproche plus

oyer-Moore echerche à l'envers

Décalages par sauts Prétraitement du motif

orithme de Boyer-Moor nplifié - version rspool)

Complexité



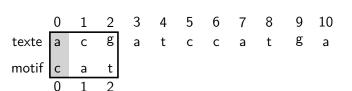


FIGURE 2 – Première comparaison : pas de correspondance

Décalage de la fenêtre

Recherche textuelle

Principe

FIGURE 3 – Première comparaison : correspondance

Recherche textuelle

roblématique

Principe 113

Principe

Approche plus efficace :

Recherche à l'envers Décalages par sauts Prétraitement du motif

etraitement du motif corithme de Boyer-Moor nplifié - version rspool)

Complexité

FIGURE 4 – Deuxième comparaison : pas de correspondance

Décalage de la fenêtre

Recherche textuelle

Principe

10 / 49



Sommaire

robiematique

2. Approche naïve

2.1 Principe

2.2 Implémentation

3 Approche plus efficace : Boyer 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

1 of measure_name(near size, each size) \rightarrow set

1 of measure_name(near size, each size) \rightarrow set

1 of measure_name(near size, each size size, each size,
```

Correction

```
def recherche naive(texte: str, motif: str) -> int:
 3
       renvoie la position du motif dans le texte
       −1 s'il n'est pas présent
 4
 5
       # dernière position = taille(texte) - taille(motif)
 6
       for i in range(len(texte)—len(motif)+1):
 8
          i = 0
          while (i < len(motif)) and (motif[i] == texte[i+j]):
10
             i += 1
          if j == len(motif): # correspondance sur toute la fenê
11
        tre
12
              return i
13
       return -1
```

Code 1 – Approche naïve

Recherche textuelle

Problématique

oproche naïve

Implémentation

proche plus cace :

cherche à l'envers calages par sauts étraitement du motif

traitement du motif orithme de Boyer-Mooi nplifié - version

omplexité

O(P.S)

Imaginors le cas La complexité dépend de la taille de la fenêtre et de celle du motif.

Correction

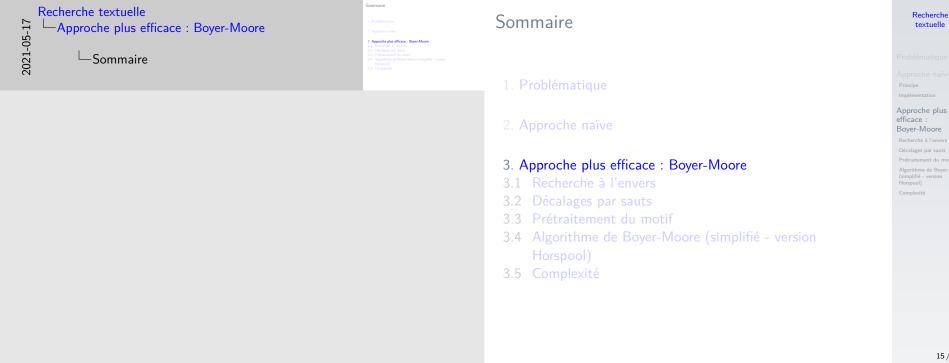
Imaginons le cas :

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

Recherche

textuelle

Implémentation



15 / 49

Recherche textuelle
Approche plus efficace : Boyer-Moore

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

Bover-Moore

Boyer-Moore

└─Boyer-Moore

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

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

blematique

Recherche

textuelle

Approche naïve
Principe

Implémentation

Approche plus efficace :
Boyer-Moore

echerche à l'envers

calages par sauts traitement du motif

thme de Boyer-Moord lifié - version ool)

nplexité

► 1970 : algorithme de Knuth-Morris-Pratt. O(T + M) ► 1977 : algorithme de Bover-Moore.

Bover-Moore

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

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

▶ 1977 : algorithme de Boyer-Moore.

Recherche

textuelle

Approche plus

efficace: Boyer-Moore

Bover-Moore

- 1. évoguera 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.

▶ 1980 : Horspool propose une version simplifiée de l'algorithme de Boyer-Moore. *O*(*T*)

Recherche textuelle

Problematique

Principe

Approche plus

efficace : Boyer-Moore

Recherche à l'envers

Prétraitement du motif

orithme de Boyer-Moor nplifié - version rspool)

plexité



Recherche à l'envers

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

									8		
texte	а	С	g	а	t	С	С	а	t	g	а
texte motif	С	a	t								
	0	1	2								

FIGURE 5 – Première comparaison : pas de correspondance

Recherche textuelle

Problematique

Principe

Approche plus efficace :

Recherche à l'envers

Décalages par sauts

lgorithme de Boyer-Moore implifié - version

Complexité

Recherche textuelle

Approche plus efficace : Boyer-Moore

Recherche à l'envers

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

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

oche naïve

Recherche

textuelle

Principe Implémentation

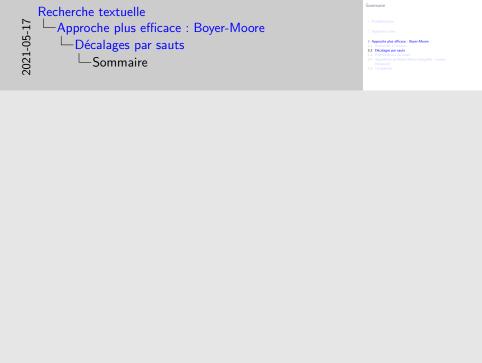
proche plus

Boyer-Moore Recherche à l'envers

echerche à l'envers Vécalages par sauts

> traitement du motif orithme de Boyer-Moo oplifié - version

> > exité



Sommaire

- 3. Approche plus efficace : Boyer-Moore
- 3.2 Décalages par sauts

- 3.4 Algorithme de Boyer-Moore (simplifié version

Recherche

textuelle

- Décalages par sauts

Recherche textuelle -Approche plus efficace : Boyer-Moore -Décalages par sauts -Décalage par sauts

Décalage par sauts Le motif ne contient pas la lettre g (la dernière lettre de la atccat ga FIGURE 6 - Comparaisons inutiles

Décalage par sauts

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

FIGURE 6 – Comparaisons inutiles

Recherche textuelle

Décalages par sauts

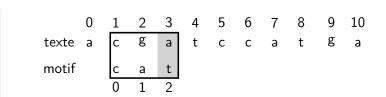


FIGURE 7 – Comparaison inutile

olématique

nroche naïve

Recherche

textuelle

Principe

Approche plu

Boyer-Moore

Décalages par sauts

Decalages par sauts
Prétraitement du motif
Algorithme de Boyer-Moor

orithme de Boyer-Moor nplifié - version rspool)

omplexité



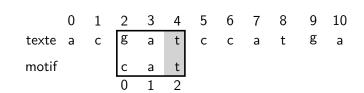


FIGURE 8 – Comparaison inutile

textuelle

Recherche

robiematique

Approche naïve Principe

Approche plus efficace :

Boyer-Moore Recherche à l'envers

Décalages par sauts Prétraitement du motif

étraitement du motif gorithme de Boyer-Moor mplifié - version irspool)

omplexité

Recherche textuelle -Approche plus efficace : Boyer-Moore └─Décalages par sauts



On peut donc directement décaler le motif à l'indice 3 du texte (figure 9).

FIGURE 9 – Décalage par saut

Recherche textuelle

Décalages par sauts

Recherche textuelle Approche plus efficace : Boyer-Moore Décalages par sauts



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

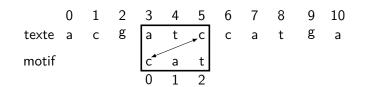


FIGURE 10 – Nouvelle situation

Recherche textuelle

Problématique

Approche naïve Principe

Implémentation

Approche plus

ficace :

oyer-Moore

echerche à l'envers

Décalages par sauts Prétraitement du motif

Algorithme de Boyer-Moor (simplifié - version Horspool)

mplexité

Recherche textuelle Approche plus efficace : Boyer-Moore Décalages par sauts



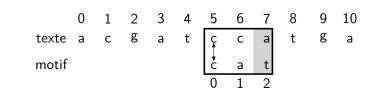


FIGURE 11 – Décalage par saut

Recherche textuelle

Problématique

Approche naïve Principe

Implémentation

efficace :
Boyer-Moore

Recherche à l'envers Décalages par sauts

Prétraitement du motif

gorithme de Boyer-Moor mplifié - version rspool)



- 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

Problématique

Principe

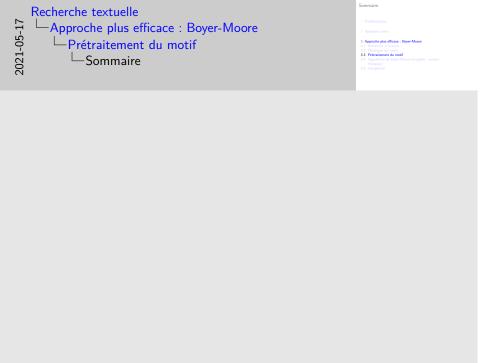
Implémentation

cace : yer-Moore

Décalages par sauts

étraitement du motif

gorithme de Boyer-Moore implifié - version orspool)



Sommaire

- 3. Approche plus efficace : Boyer-Moore

- 3.3 Prétraitement du motif

- 3.4 Algorithme de Boyer-Moore (simplifié version

Recherche

textuelle

Prétraitement du motif

Recherche textuelle

Approche plus efficace : Boyer-Moore

Prétraitement du motif

Prétraitement du motif

Pour pouvoir déclaire par saux, il faux connaître la demission de chaques lettre dans le modif. Le prétraitement consiste à calciule le déclaige à appliquer pour annexe textes a c 8 faire de la faire de déclaire conscitée à desire de la faire de

Prétraitement du motif

Prétraitement du motif

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.

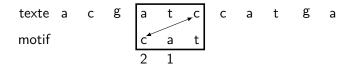


FIGURE 12 – Calculs des décalages

Recherche textuelle

Problématique

Principe

Implémentation

Approche plus efficace :

Bover-Moore

Recherche à l'envers

Prétraitement du motif

orithme de Boyer-Moor nplifié - version rspool)

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.

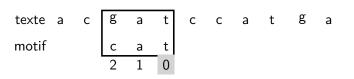


FIGURE 13 – Sauf la dernière lettre

Recherche textuelle

Problématique

pproche naive Principe

fficace :
Soyer-Moore

Décalages par sauts

Prétraitement du motif

Algorithme de Boyer-Moo simplifié - version Horspool)

Complexité

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

c g a t c c a t g a

Autre exemple

texte motif 2

FIGURE 14 - Répétition dans le motif

Recherche textuelle

Prétraitement du motif

Activité 2 : Écrire la fonction pretraitement_decalages(motif: atr) \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

Problématique

Principe

Approche plus efficace :

Recherche à l'envers
Décalages par sauts
Prétraitement du motif

corithme de Boyer-Moor mplifié - version rspool) S'il y a répétition (slide précédent) le dictionnaire est mis à jour (ligne 10).

Correction

def pretraitement decalages(motif: str) -> dict: renvoie le dictionnaire des décalages à appliquer 3 pour chaque lettre du motif (sauf dernière) 5 decalages = dict()6 # on s'arrête à l'avant dernière lettre du motif for i in range(len(motif)-1): # len(motif)—1 est la position de la dernière lettre decalages[motif[i]] = len(motif)-1-i10 return decalages 11

cherche à l'envers calages par sauts

Prétraitement du motif

thme de Boyer-Moor ifié - version ool)



Sommaire

- 3. Approche plus efficace : Boyer-Moore

- 3.4 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 Boyer-Moore
L'algorithme de Boyer-Moore v'écrit alon :

1 Giver to talleure de décision s'écrit alon :

1 Giver to talleure des décisions s'écrit alon :

1 Test qu'er le vier par à la fin de teste

2 Test qu'er le vier par à la fin de teste

1 Service par à la fin de teste

2 Brancy par alon partiers

2 Brancy parailleur

3 Brancy parailleur

4 Brancy L'algorithme de Boyer-Moore (version Horspool)

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 2 – Algorithme de Boyer-Moore (version Horspool)

Recherche textuelle

Problématique

Principe

pproche plus ficace :

Recherche à l'envers

Prétraitement du motif

Algorithme de Boyer-Moore (simplifié - version Horspool)

mplexité

Recherche textuelle

Approche plus efficace : Boyer-Moore

Algorithme de Boyer-Moore (simplifié - version Horspool)

:tivité 3 : L'Écrire la fonction compare(texte: str, position: int, motif: str) → bool qui remvoie True si le motif est présent à la position i

renvoie *True* si le motif est présent à la posit du texte. Écrire la fonction

decalage_fenetre(decalages: dict, taille: int, lettre: str) → int qui remvois le décalage à appliquer pour faire coincider le motif à la dermière lettre de la fenêtre. Si la lettre n'est pas présente, la taille du motif est

Ecrire alors la fonction boyer_moore (texte: str, motif: str)
int qui renvoie la 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.

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.

Recherche textuelle

Problématique

Approche naïve

Implémentation

Approche plus efficace :

Recherche à l'envers

Décalages par sauts Prétraitement du motif

Algorithme de Boyer-Moore (simplifié - version Horspool)

.omplexite

```
Correction

of competition in, notif on ) > bool

of competition in, notif on ) > bool

of competition in the control of the control

of control of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control

of control
```

Correction

```
def compare(texte: str, position: int, motif: str) -> bool:
       compare le morceau du texte
       (en partant de position + taille(motif))
 4
       avec le motif
 5
 6
       Returns:
           bool: True si on a trouvé le motif
       77 77 77
       # position de la dernière lettre de la fenêtre
10
       en cours = position+len(motif)-1
11
       # parcours de la fenêtre à l'envers
12
       for i in range(len(motif)-1, -1, -1):
13
          if not(texte[en_cours] == motif[i]):
14
              return False
15
           else:
16
              en_cours -= 1
17
       return True
18
```

Recherche textuelle

oblématique

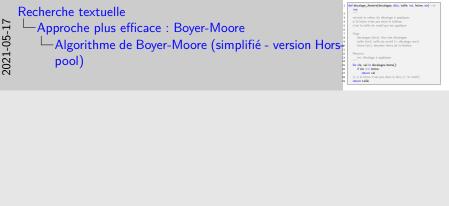
Principe

proche plus

echerche à l'envers

Prétraitement du motif

Algorithme de Boyer-Moore
(simplifié - version
Horspool)



```
int:
77 77 77
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
```

def decalage_fenetre(decalages: dict, taille: int, lettre: str) ->

Recherche

textuelle

Algorithme de Boyer-Moore (simplifié - version

Horspool)

```
Recherche textuelle

Approche plus efficace : Boyer-Moore

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

```
and decoding—interesting and its table int, lattice on)—interest in the control of the control o
```

```
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
```

Recherche textuelle

Problèmatique

Principe 11a1

pproche plus ficace :

Recherche à l'envers Décalages par sauts Prétraitement du motif

Algorithme de Boyer-Moore (simplifié - version Horspool)



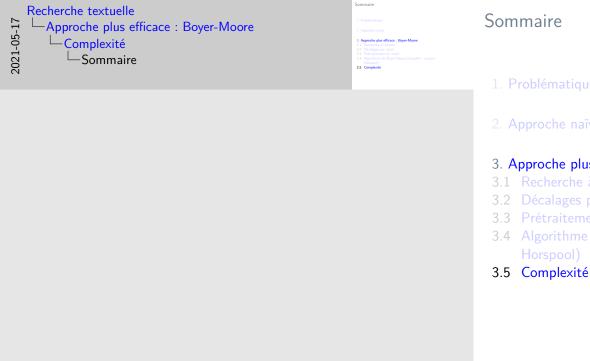
```
def boyer_moore(texte: str, motif: str) -> int:
 3
        Returns:
          int: la position du motif dans le texte, -1 sinon.
 5
       decalages = pretraitement_decalages(motif)
 6
       i = 0
       while i \le len(texte) - len(motif):
 8
           # si on trouve le motif
           if compare(texte,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

Recherche

textuelle

Algorithme de Boyer-Moore (simplifié - version

Horspool)



- 3. Approche plus efficace : Boyer-Moore

- 3.4 Algorithme de Boyer-Moore (simplifié version

Complexité

Recherche

textuelle

40 / 49

Complexité

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

a a a b a a a b a a a b

FIGURE 15 – Un cas représentatif

c c c c c

Problématique

Recherche

textuelle

.......

Principe Principe

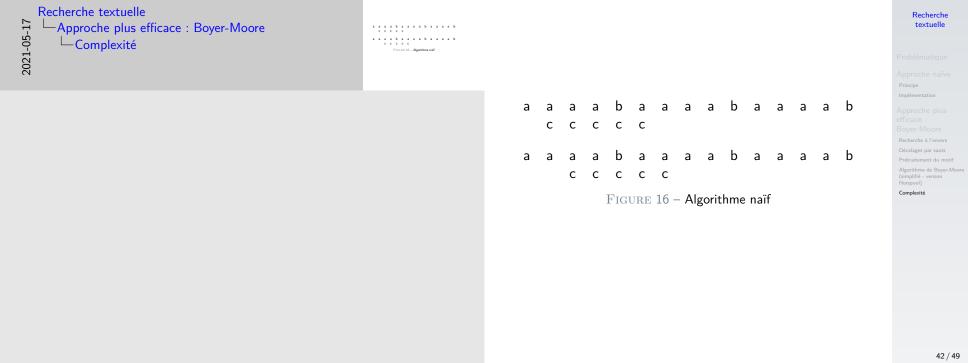
Implémentation

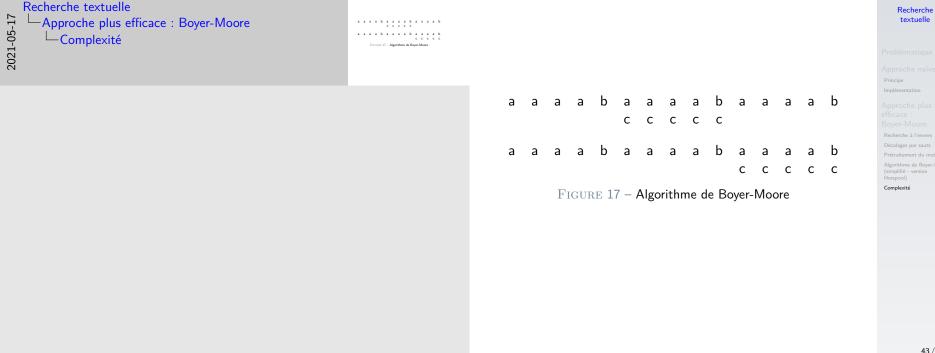
pproche plus ficace : oyer-Moore

cherche à l'envers calages par sauts

traitement du motif corithme de Boyer-Moor

orithme de Boyer-Moor nplifié - version rspool)





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

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

phlématique

Recherche

textuelle

obiemanque

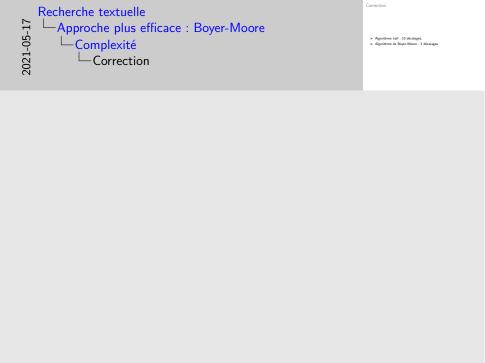
Principe

Implémentation

cace : /er-Moore

ecalages par sauts étraitement du motif

aitement du motif ithme de Boyer-Moore difié - version pool)



Correction

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

Recherche

textuelle

Recherche textuelle

Approche plus efficace : Boyer-Moore
Complexité

Remarques

> Dans le meilleur des cas, la complexité temposelle de l'algorithme est 0/17/M où T est la taille du totte et M celle de mocif.

> Plus le mocif est long plus l'algorithme est rapide.

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

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.

Recherche textuelle

Problèmatique

Principe

Implémentation

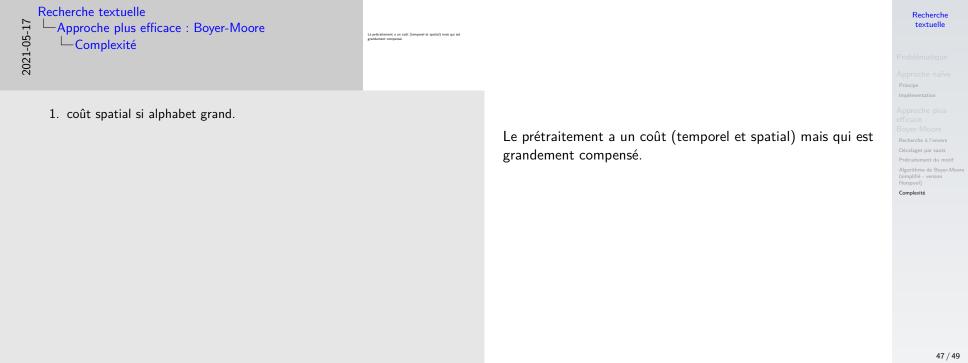
pproche plus fficace : oyer-Moore

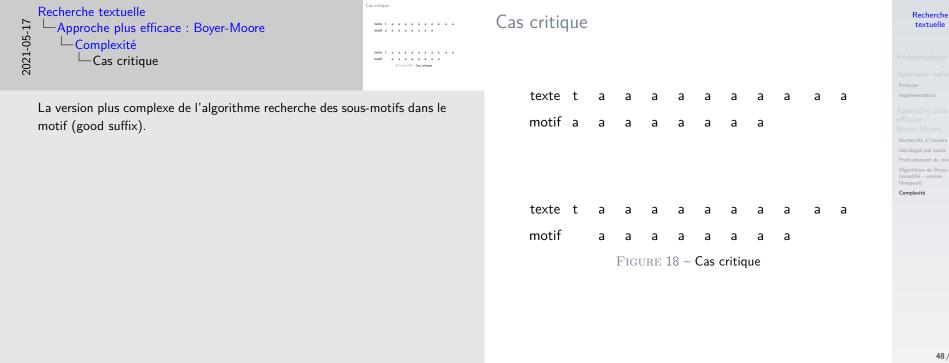
Recherche à l'envers

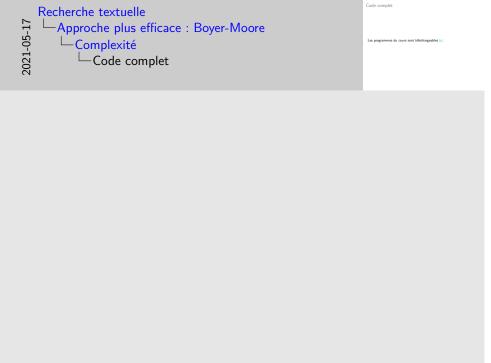
Décalages par sauts

Prétraitement du motif

rétraitement du motif gorithme de Boyer-Moore implifié - version orspool)







Code complet

Les programmes du cours sont téléchargeables ici.

Recherche

textuelle