Recherche exacte de motif

Algotihmes naïf, MP et KMP

Sèverine Bérard

Institut des Sciences de l'Evolution-Montpellier

ISE-M – FDS, Université de Montpellier

Plan

- Introduction
- Algorithme naïf
- Algorithme MP
- Algorithme KMP
- Références

- Introduction
- Algorithme naïf
- Algorithme MP
- Algorithme KMP
- Références

Bord et période (encore)

- Soit u un mot non vide, p un entier tel que $0 < p \leqslant |u|$
- p est une période de u si une de ces conditions est satisfaite :

Bord et période (encore)

- Soit u un mot non vide, p un entier tel que $0 < p \leqslant |u|$
- p est une période de u si une de ces conditions est satisfaite :

   1. $u[i] = u[i + p]$, pour $1 \leqslant i \leqslant |u| - p$

Bord et période (encore)

- Soit u un mot non vide, p un entier tel que $0 < p \leqslant |u|$
- p est une période de u si une de ces conditions est satisfaite :

1. $u[i] = u[i + p]$, pour $1 \leqslant i \leqslant |u| - p$
2. u est un préfixe d'un mot y^k, $k > 0$, $|y| = p$

Bord et période (encore)

- Soit u un mot non vide, p un entier tel que $0 < p \leqslant |u|$
- p est une période de u si une de ces conditions est satisfaite :

1. $u[i] = u[i+p]$, pour $1 \leqslant i \leqslant |u| - p$
2. u est un préfixe d'un mot y^k, $k > 0$, $|y| = p$
3. $u = yw = wz$, pour des mots y, z et w avec $|y| = |z| = p$
   Le mot w est appelé bord de u

Bord et période (encore)

- Soit u un mot non vide, p un entier tel que $0 < p \leqslant |u|$
- p est une période de u si une de ces conditions est satisfaite :

  1. $u[i] = u[i+p]$, pour $1 \leqslant i \leqslant |u| - p$
  2. u est un préfixe d'un mot y^k, $k > 0$, $|y| = p$
  3. $u = yw = wz$, pour des mots y, z et w avec $|y| = |z| = p$
     Le mot w est appelé bord de u

- $period(u)$ est la plus petite période de u (peut être $|u|$)

Bord et période (encore)

- Soit u un mot non vide, p un entier tel que $0 < p \leqslant |u|$
- p est une période de u si une de ces conditions est satisfaite :

  1. $u[i] = u[i+p]$, pour $1 \leqslant i \leqslant |u| - p$
  2. u est un préfixe d'un mot y^k, $k > 0$, $|y| = p$
  3. $u = yw = wz$, pour des mots y, z et w avec $|y| = |z| = p$
     Le mot w est appelé bord de u

- $period(u)$ est la plus petite période de u (peut être $|u|$)
- $border(u)$ est le plus long bord de u (peut être ϵ)

Bord et période (encore)

- Soit u un mot non vide, p un entier tel que $0 < p \leqslant |u|$
- p est une période de u si une de ces conditions est satisfaite :

  1. $u[i] = u[i+p]$, pour $1 \leqslant i \leqslant |u| - p$
  2. u est un préfixe d'un mot y^k, $k > 0$, $|y| = p$
  3. $u = yw = wz$, pour des mots y, z et w avec $|y| = |z| = p$

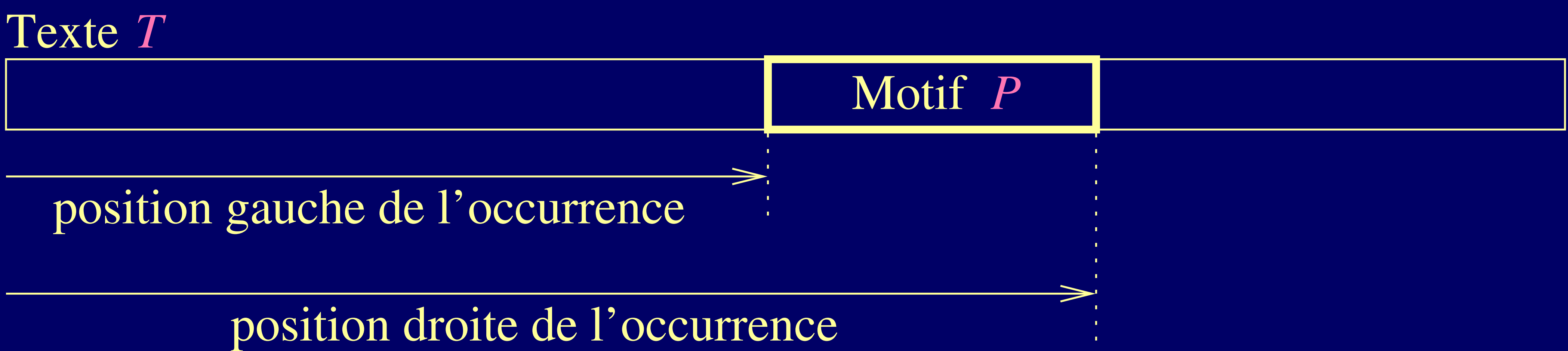
     Le mot w est appelé bord de u

- $period(u)$ est la plus petite période de u (peut être $|u|$)
- $border(u)$ est le plus long bord de u (peut être ϵ)

4	$abacaba$
8	aba
10	a
11	ϵ

Périodes et bords du mot $abacabacaba$ de longueur 11

Trouver toutes les occurrences d'un motif P de longueur m à l'intérieur d'un texte T de longueur n



Recherche de motif exact : exemple

- Motif : un mot P de longueur m

  t a t a

Recherche de motif exact : exemple

- Motif : un mot P de longueur m

  t a t a

- Texte : un mot T de longueur n

  a g g c t c a c g t a t a t a t g c g t t a t a a t

Recherche de motif exact : exemple

- **Motif** : un mot P de longueur m

  t a t a

- **Texte** : un mot T de longueur n

  a g g c t c a c g t a t a t a t g c g t t a t a a t

- **Occurrences** :

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
a	g	g	c	t	c	a	c	g	t	a	t	a	t	a	t	g	c	g	t	t	a	t	a	a	t
									t	a	t	a								t	a	t	a		
											t	a	t	a											

Recherche de motif exact : exemple

- Motif : un mot P de longueur m

  t a t a

- Texte : un mot T de longueur n

  a g g c t c a c g t a t a t a t g c g t t a t a a t

- Occurrences :

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
a	g	g	c	t	c	a	c	g	t	a	t	a	t	a	t	g	c	g	t	t	a	t	a	a	t
									t	a	t	a								t	a	t	a		
											t	a	t	a											

Le motif P apparaît aux positions 10, 12 et 21 dans le texte T

Recherche de motif exact : exemple

- Motif : un mot P de longueur m

  t a t a

- Texte : un mot T de longueur n

  a g g c t c a c g t a t a t a t g c g t t a t a a t

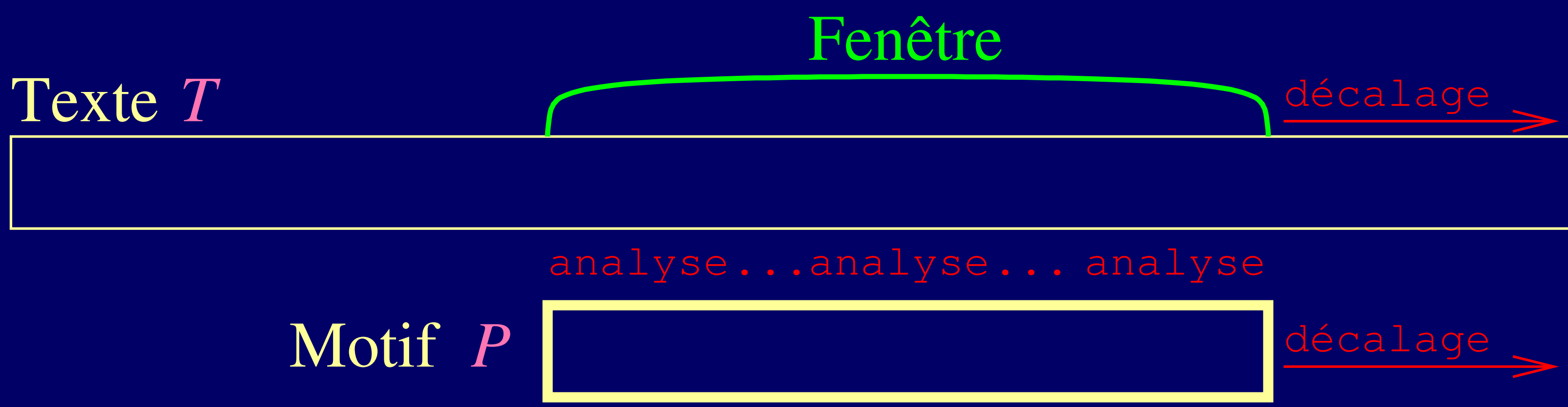
- Occurrences :

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
a	g	g	c	t	c	a	c	g	t	a	t	a	t	a	t	g	c	g	t	t	a	t	a	a	t
									t	a	t	a								t	a	t	a		
											t	a	t	a											

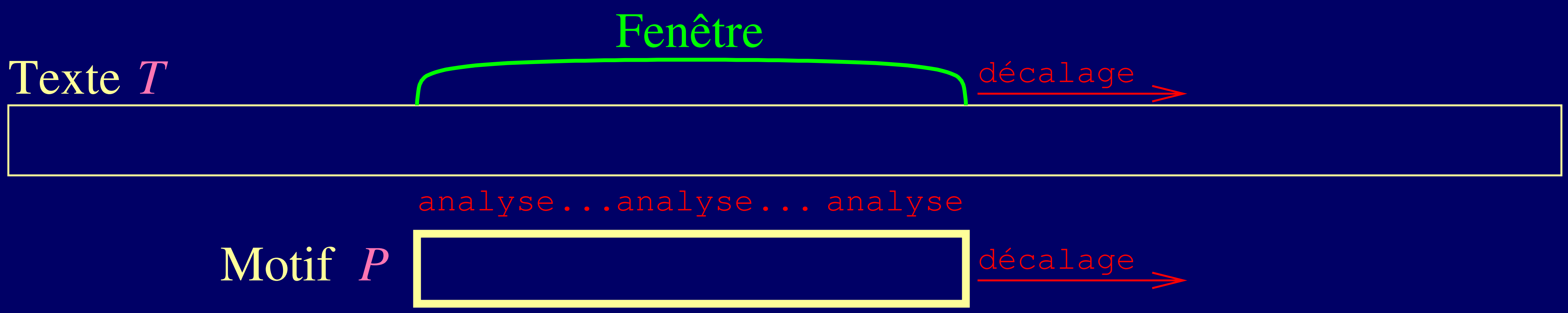
Le motif P apparaît aux positions 10, 12 et 21 dans le texte T

- Opérations élémentaires : comparaison de symboles ($=$,$\neq$)

Stratégie de la fenêtre glissante



Stratégie de la fenêtre glissante



Algorithme 1 : Mécanisme « analyse et décalage »

Données : Deux chaînes T et P de longueurs respectives n et m.

Placer la Fenêtre au début du texte ;

tant que la Fenêtre est sur le texte **faire**

  analyse : **si** Fenêtre = Motif **alors** le rapporter ;

  décalage : déplacer la Fenêtre vers la droite et

  mémoriser des informations à utiliser durant les prochaines analyses et décalages ;

- Introduction
- Algorithme naïf
- Algorithme MP
- Algorithme KMP
- Références

Algorithme naïf

- Principes : Pas de mémorisation, décalage d'une seule position
- Complexité : $O(m \times n)$

Algorithme 2 : Recherche naïve

Données : Deux chaînes T et P de longueurs respectives n et m.

$pos := 1$;

tant que $pos \leqslant n - m + 1$ **faire**

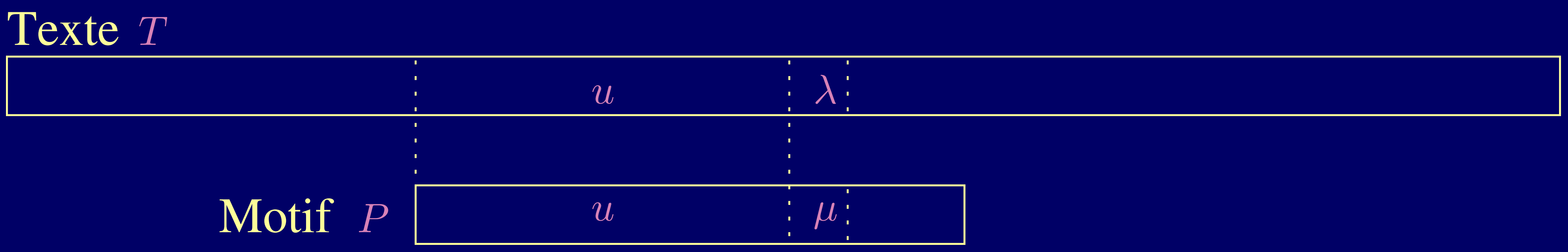
  $i := 1$;

  tant que $(i \leqslant m$ **et** $P[i] = T[pos + i - 1])$ **faire** $i := i + 1$;

  si $i = m + 1$ **alors**

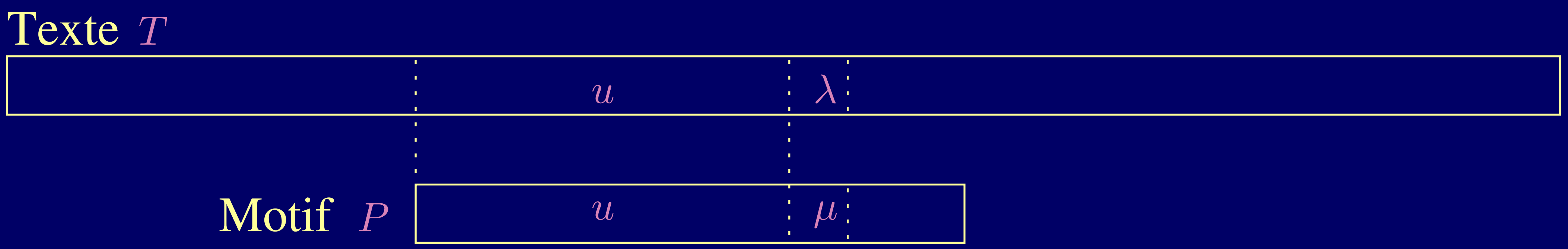
    Écrire("P apparaît à la position ", pos) ;

  $pos := pos + 1$;



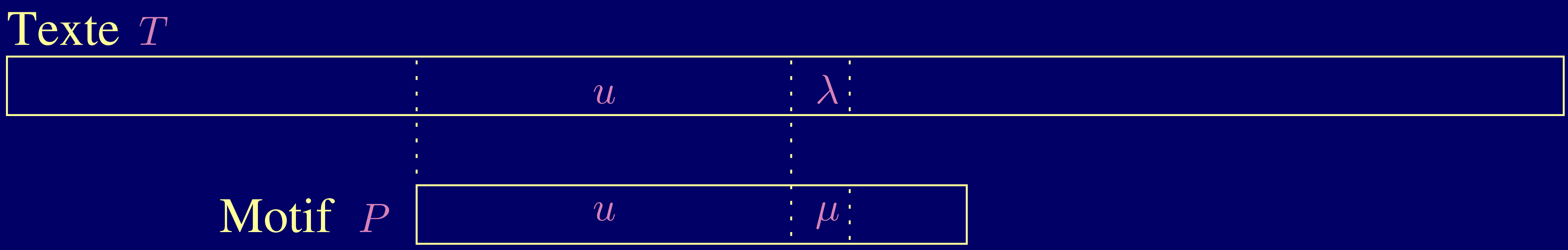
Texte T
u
λ
Motif P
u
μ

Décalage plus judicieux ?



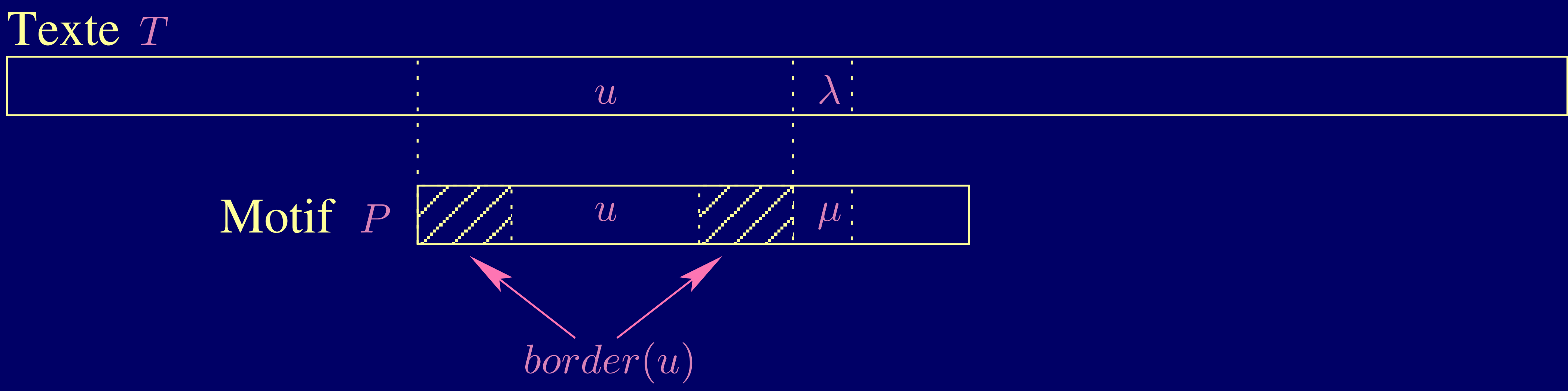
- Situation d'échec : identité des caractères de u mais $\lambda \neq \mu$

Décalage plus judicieux ?



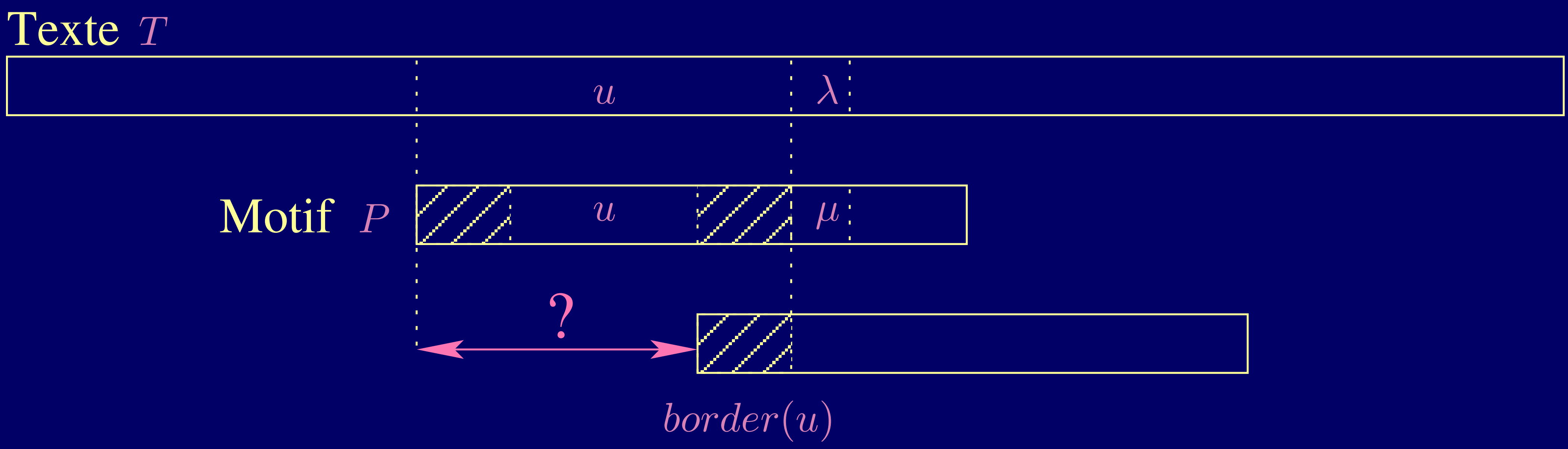
- Situation d'échec : identité des caractères de u mais $\lambda \neq \mu$
- u est un préfixe de P

Décalage plus judicieux ?



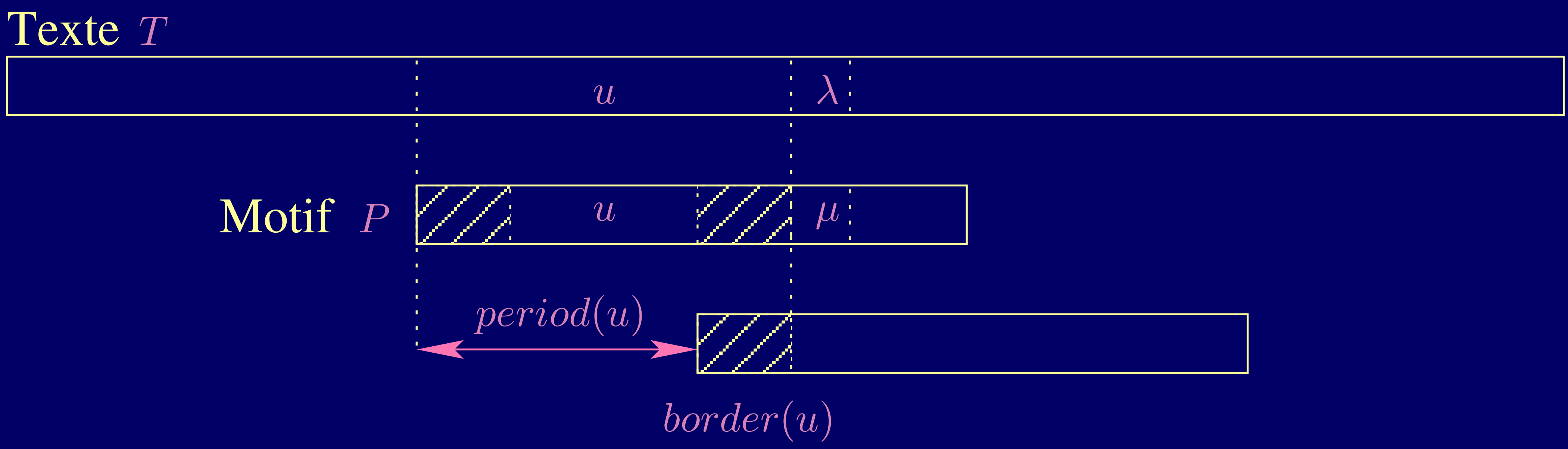
- Situation d'échec : identité des caractères de u mais $\lambda \neq \mu$
- u est un préfixe de P

Décalage plus judicieux ?



- Situation d'échec : identité des caractères de u mais $\lambda \neq \mu$
- u est un préfixe de P

Décalage plus judicieux ?



- Situation d'échec : identité des caractères de u mais $\lambda \neq \mu$
- u est un préfixe de P
- $|u| - |border(u)| = period(u)$

- Introduction
- Algorithme naïf
- Algorithme MP
- Algorithme KMP
- Références

Stratégie améliorée

- Longueur du décalage ≥ 1 : taille de la période $period(u)$
- Mémorisation des bords

Stratégie améliorée

- Longueur du décalage ≥ 1 : taille de la période $period(u)$
- Mémorisation des bords

Algorithme 3 : Mécanisme « analyse et décalage » amélioré

Données : Deux chaînes T et P de longueurs respectives n et m.

Placer la Fenêtre au début du texte ;

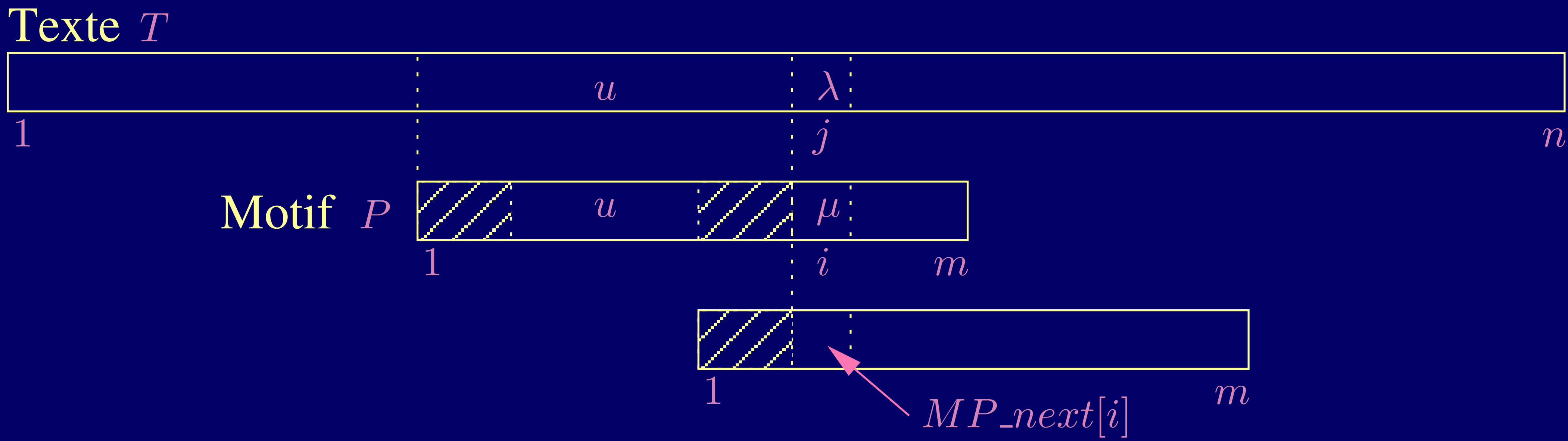
tant que la Fenêtre est sur le texte **faire**

  u := plus long préfixe commun entre la Fenêtre et le Motif ;

  si u = Motif **alors** le rapporter ;

  Décaler la Fenêtre de $period(u)$ places vers la droite ;

  Mémoriser $border(u)$;


Texte T
u
λ
1
j
n
Motif P
u
μ
1
i
m
1
m
$MP\_next[i]$

Algorithme de Morris-Pratt (1970)

Algorithme 4 : Algorithme MP

Données : Deux chaînes T et P de longueurs respectives n et m.

$i := 1$; $j := 1$;

tant que $j \leqslant n$ **faire**

  tant que $(i > 0$ **et** $P[i] \neq T[j])$ **faire**

    $i := MP\_next[i]$;

  $i := i + 1$; $j := j + 1$;

  si $i = m + 1$ **alors**

    Écrire(“P apparaît à la position ”, $j - i + 1$) ;

    $i := MP\_next[i]$;

Calcul des bords des préfixes de P

- Remarque : un bord d'un bord de u est un bord de u

Calcul des bords des préfixes de P

- Remarque : un bord d'un bord de u est un bord de u
- On calcule un tableau $BORD$, tel que $BORD[i] = |border(P[1..i])|$

Calcul des bords des préfixes de P

- Remarque : un bord d'un bord de u est un bord de u
- On calcule un tableau $BORD$, tel que $BORD[i] = |border(P[1..i])|$

Algorithme 5 : Calcule $BORD$

Données : Un mot P de longueur m

$BORD[0] := -1$;

pour (i de 1 à m) **faire**

  $j := BORD[i-1]$;

  tant que ($j \geqslant 0$ **et** $P[i] \neq P[j+1]$) **faire** $j := BORD[j]$;

  $BORD[i] := j+1$;

Calcul des bords des préfixes de P

- Remarque : un bord d'un bord de u est un bord de u
- On calcule un tableau $BORD$, tel que $BORD[i] = |border(P[1..i])|$

Algorithme 5 : Calcule $BORD$

Données : Un mot P de longueur m

$BORD[0] := -1$;

pour (i de 1 à m) **faire**

  $j := BORD[i-1]$;

  tant que ($j \geqslant 0$ **et** $P[i] \neq P[j+1]$) **faire** $j := BORD[j]$;

  $BORD[i] := j+1$;

- i parcourt les longueurs des préfixes de manière croissante

Calcul des bords des préfixes de P

- Remarque : un bord d'un bord de u est un bord de u
- On calcule un tableau $BORD$, tel que $BORD[i] = |border(P[1..i])|$

Algorithme 5 : Calcule $BORD$

Données : Un mot P de longueur m

$BORD[0] := -1$;

pour (i de 1 à m) **faire**

  $j := BORD[i-1]$;

  tant que ($j \geqslant 0$ **et** $P[i] \neq P[j+1]$) **faire** $j := BORD[j]$;

  $BORD[i] := j + 1$;

- i parcourt les longueurs des préfixes de manière croissante
- j parcourt les longueurs des bords de manière décroissante

Pré-traitement de l'algorithme MP : MP_next

Remarque :

$$MP\_next[i] =$$

Remarque :

$MP\_next[i] = BORD[i-1]+1$, pour i de 1 à $m+1$

Remarque :

$MP\_next[i] = BORD[i-1]+1$, pour i de 1 à $m+1$

$\Rightarrow$ Quasiment même algorithme

Pré-traitement de l'algorithme MP : MP_next

Remarque :

$MP\_next[i] = BORD[i-1]+1$, pour i de 1 à $m+1$

$\Rightarrow$ Quasiment même algorithme

Algorithme 6 : Calcule MP_next

Données : Un mot P de longueur m

```
MP_next[1] := 0 ; j := 0 ;
pour (i de 1 à m) faire
    /* À chaque entrée de boucle on a j := MP_next[i]          */
    tant que (j > 0 et P[i] ≠ P[j]) faire j := MP_next[j] ;
    j := j + 1 ;
    MP_next[i + 1] := j ;
```

Remarque :

$$MP\_next[i] = BORD[i-1]+1, \text{ pour } i \text{ de } 1 \text{ à } m+1$$

$\Rightarrow$ Quasiment même algorithme

Algorithme 6 : Calcule MP_next

Données : Un mot P de longueur m

$MP\_next[1] := 0$; $j := 0$;

pour (i de 1 à m) **faire**

  /* À chaque entrée de boucle on a $j := MP\_next[i]$ */

  tant que ($j > 0$ **et** $P[i] \neq P[j]$) **faire** $j := MP\_next[j]$;

  $j := j + 1$;

  $MP\_next[i+1] := j$;

- i parcourt les longueurs des préfixes de manière croissante

Remarque :

$MP\_next[i] = BORD[i-1]+1$, pour i de 1 à $m+1$

$\Rightarrow$ Quasiment même algorithme

Algorithme 6 : Calcule MP_next

Données : Un mot P de longueur m

$MP\_next[1] := 0$; $j := 0$;

pour (i de 1 à m) **faire**

```
/* À chaque entrée de boucle on a j := MP_next[i]    */
```

tant que ($j > 0$ **et** $P[i] \neq P[j]$) **faire** $j := MP\_next[j]$;

$j := j+1$;

$MP\_next[i+1] := j$;

- i parcourt les longueurs des préfixes de manière croissante
- $j-1$ parcourt les longueurs des bords de manière décroissante

Conclusion sur l'algorithme MP

- Complexité :

Conclusion sur l'algorithme MP

- Complexité :

  1. Pré-traitement : $O(m)$ en temps et en espace

Conclusion sur l'algorithme MP

- Complexité :

  1. Pré-traitement : $O(m)$ en temps et en espace
  2. Phase de recherche : $O(n+m)$ en temps et en espace

Conclusion sur l'algorithme MP

- Complexité :

  1. Pré-traitement : $O(m)$ en temps et en espace
  2. Phase de recherche : $O(n+m)$ en temps et en espace

- Une fois le pré-traitement effectué sur un motif, on peut le rechercher dans autant de textes que souhaités

Conclusion sur l'algorithme MP

- Complexité :
  1. Pré-traitement : $O(m)$ en temps et en espace
  2. Phase de recherche : $O(n+m)$ en temps et en espace
- Une fois le pré-traitement effectué sur un motif, on peut le rechercher dans autant de textes que souhaités
- Intérêts :
  1. Phase de recherche plus rapide que l'algorithme naïf

Conclusion sur l'algorithme MP

- Complexité :
  1. Pré-traitement : $O(m)$ en temps et en espace
  2. Phase de recherche : $O(n+m)$ en temps et en espace
- Une fois le pré-traitement effectué sur un motif, on peut le rechercher dans autant de textes que souhaités
- Intérêts :
  1. Phase de recherche plus rapide que l'algorithme naïf
  2. Très facile à implémenter

Conclusion sur l'algorithme MP

- Complexité :
  1. Pré-traitement : $O(m)$ en temps et en espace
  2. Phase de recherche : $O(n+m)$ en temps et en espace
- Une fois le pré-traitement effectué sur un motif, on peut le rechercher dans autant de textes que souhaités
- Intérêts :
  1. Phase de recherche plus rapide que l'algorithme naïf
  2. Très facile à implémenter
- Inconvénients : besoin de temps et d'espace supplémentaire ($O(m)$)

- Complexité :
  1. Pré-traitement : $O(m)$ en temps et en espace
  2. Phase de recherche : $O(n+m)$ en temps et en espace
- Une fois le pré-traitement effectué sur un motif, on peut le rechercher dans autant de textes que souhaités
- Intérêts :
  1. Phase de recherche plus rapide que l'algorithme naïf
  2. Très facile à implémenter
- Inconvénients : besoin de temps et d'espace supplémentaire ($O(m)$)

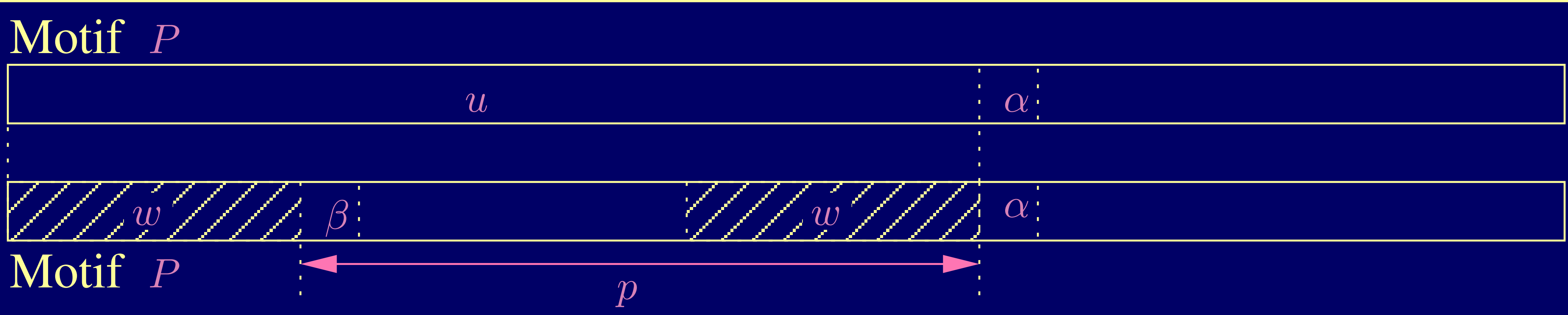
  Peut-on faire mieux ?

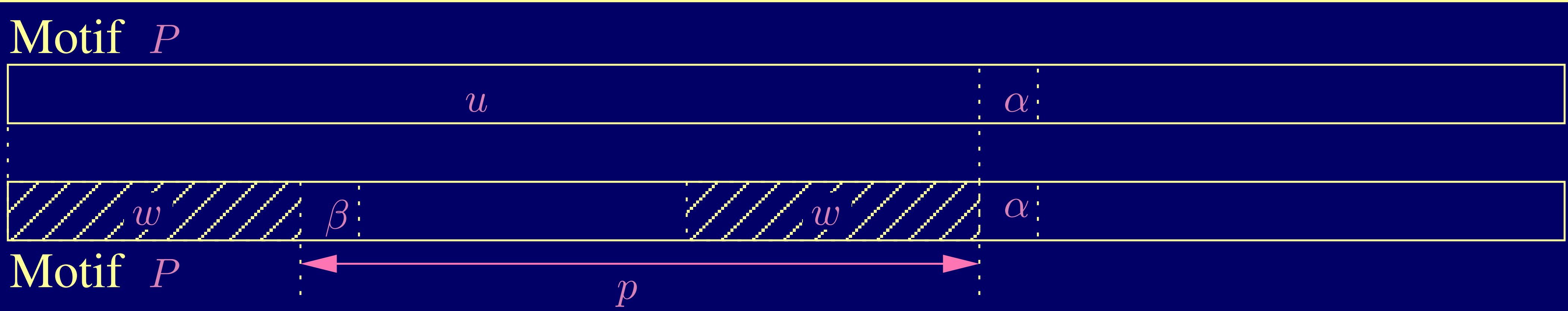
Conclusion sur l'algorithme MP

- Complexité :
  1. Pré-traitement : $O(m)$ en temps et en espace
  2. Phase de recherche : $O(n+m)$ en temps et en espace
- Une fois le pré-traitement effectué sur un motif, on peut le rechercher dans autant de textes que souhaités
- Intérêts :
  1. Phase de recherche plus rapide que l'algorithme naïf
  2. Très facile à implémenter
- Inconvénients : besoin de temps et d'espace supplémentaire ($O(m)$)

  Peut-on faire mieux ? OUI !

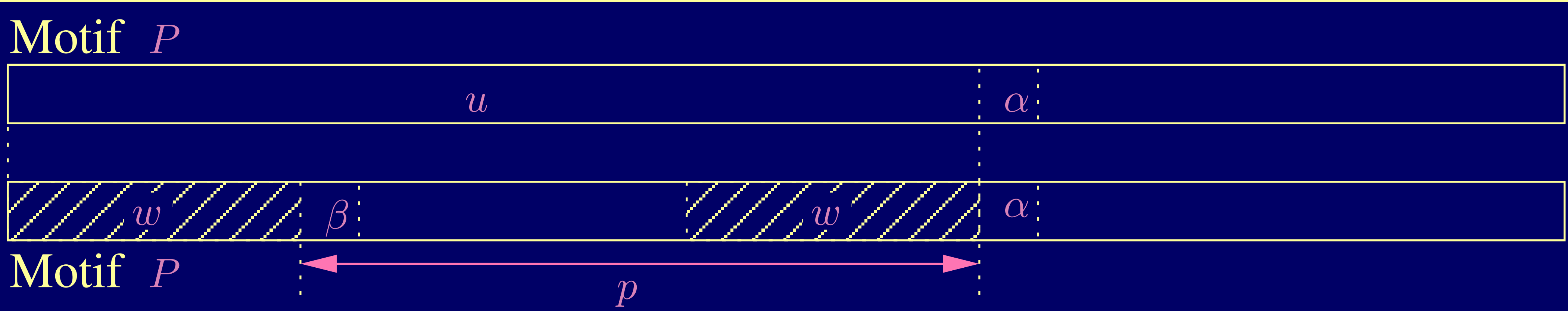
- Introduction
- Algorithme naïf
- Algorithme MP
- Algorithme KMP
- Références


Motif P
u
α
w
β
w
α
Motif P
p



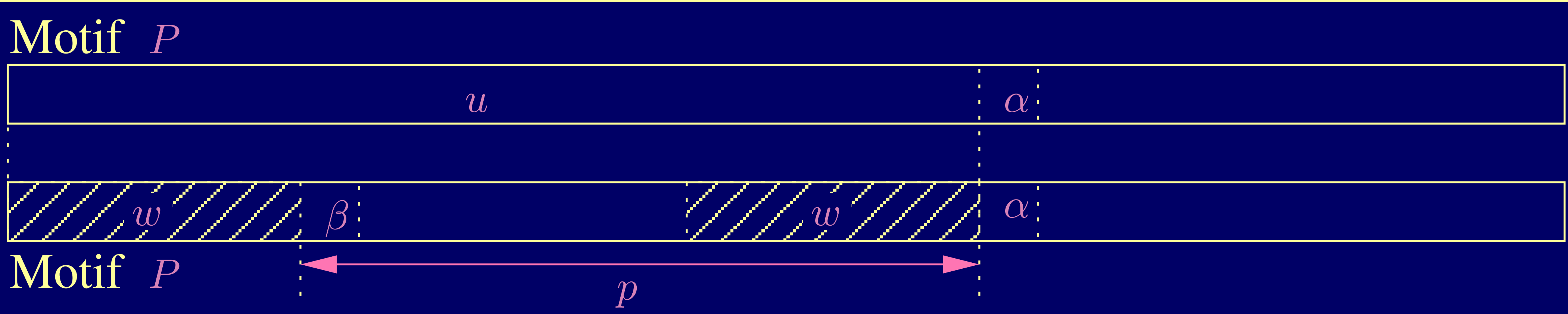
- Notions de bords stricts (w) et de périodes interrompues (p)

Amélioration de l'algorithme MP



- Notions de bords stricts (w) et de périodes interrompues (p)
- Modifie seulement la phase de pré-traitement

Amélioration de l'algorithme MP



- Notions de bords stricts (w) et de périodes interrompues (p)
- Modifie seulement la phase de pré-traitement

Algorithme 7 : Mécanisme de MP amélioré

Données : Deux chaînes T et P de longueurs respectives n et m.

Placer la Fenêtre au début du texte ;

tant que la Fenêtre est sur le texte **faire**

  $u :=$ plus long préfixe commun entre la Fenêtre et le Motif ;

  si $u =$ Motif **alors** le rapporter ;

  Décaler la Fenêtre de $\underline{period\_int(u)}$ places vers la droite ;

  Mémoriser $\underline{border\_strict(u)}$;

Bord et période (le retour)

- Soit P un mot fixé et u un préfixe non vide de P

Bord et période (le retour)

- Soit P un mot fixé et u un préfixe non vide de P
- w est un bord strict de u si à la fois :

Bord et période (le retour)

- Soit P un mot fixé et u un préfixe non vide de P

- w est un bord strict de u si à la fois :

  1. w est un bord de u

Bord et période (le retour)

- Soit P un mot fixé et u un préfixe non vide de P

- w est un bord strict de u si à la fois :

  1. w est un bord de u
  2. $w\beta$ est un préfixe de P mais pas $u\beta$

Bord et période (le retour)

- Soit P un mot fixé et u un préfixe non vide de P

- w est un bord strict de u si à la fois :

  1. w est un bord de u
  2. $w\beta$ est un préfixe de P mais pas $u\beta$

- p est une période interrompue de u si $p = |u| - |w|$ avec w un bord strict de u

Bord et période (le retour)

- Soit P un mot fixé et u un préfixe non vide de P

- w est un bord strict de u si à la fois :

  1. w est un bord de u
  2. $w\beta$ est un préfixe de P mais pas $u\beta$

- p est une période interrompue de u si $p = |u| - |w|$ avec w un bord strict de u

- Exemple : soit $u = abacabacaba$ un préfixe de $P = abacabacabacc$

Bord et période (le retour)

- Soit P un mot fixé et u un préfixe non vide de P

- w est un bord strict de u si à la fois :

  1. w est un bord de u
  2. $w\beta$ est un préfixe de P mais pas $u\beta$

- p est une période interrompue de u si $p = |u| - |w|$ avec w un bord strict de u

- Exemple : soit $u = abacabacaba$ un préfixe de $P = abacabacabacc$

10	a
11	ϵ

Périodes interrompues et bords strict de $abacabacaba$

Pré-traitement de l'algorithme KMP : KMP_next

- $k = MP\_next[i]$

Pré-traitement de l'algorithme KMP : KMP_next

- $k = MP\_next[i]$
- $KMP\_next[i] = \begin{cases} k & \text{si } P[i] \neq P[k] \text{ ou si } i = m+1 \\ KMP\_next[k] & \text{si } P[i] = P[k] \end{cases}$

- $k = MP\_next[i]$
- $KMP\_next[i] = \begin{cases} k & \text{si } P[i] \neq P[k] \text{ ou si } i = m+1 \\ KMP\_next[k] & \text{si } P[i] = P[k] \end{cases}$

Algorithme 8 : Calcule KMP_next

Données : Un mot P de longueur m

$KMP\_next[1] := 0$; $j := 0$;

pour (i de 1 à m) **faire**

  tant que ($j > 0$ **et** $P[i] \neq P[j]$) **faire** $j := KMP\_next[j]$;

  $j := j + 1$;

  si ($i = m$ **ou** $P[i+1] \neq P[j]$) **alors**

    $KMP\_next[i+1] := j$;

  sinon

    $KMP\_next[i+1] := KMP\_next[j]$

Conclusion sur l'algorithme KMP (1977)

- Complexité : idem MP pour le pire des cas

Conclusion sur l'algorithme KMP (1977)

- Complexité : idem MP pour le pire des cas

  1. Pré-traitement : $O(m)$ en temps et en espace

Conclusion sur l'algorithme KMP (1977)

- Complexité : idem MP pour le pire des cas

  1. Pré-traitement : $O(m)$ en temps et en espace

  2. Phase de recherche : $O(n+m)$ en temps et en espace

Conclusion sur l'algorithme KMP (1977)

- Complexité : idem MP pour le pire des cas
  1. Pré-traitement : $O(m)$ en temps et en espace
  2. Phase de recherche : $O(n+m)$ en temps et en espace
- Une fois le pré-traitement effectué sur un motif, on peut le rechercher dans autant de textes que souhaités

Conclusion sur l'algorithme KMP (1977)

- Complexité : idem MP pour le pire des cas

  1. Pré-traitement : $O(m)$ en temps et en espace

  2. Phase de recherche : $O(n+m)$ en temps et en espace

- Une fois le pré-traitement effectué sur un motif, on peut le rechercher dans autant de textes que souhaités

- Intérêts :

  1. Phase de recherche optimisée par rapport à MP

Conclusion sur l'algorithme KMP (1977)

- Complexité : idem MP pour le pire des cas
  1. Pré-traitement : $O(m)$ en temps et en espace
  2. Phase de recherche : $O(n+m)$ en temps et en espace
- Une fois le pré-traitement effectué sur un motif, on peut le rechercher dans autant de textes que souhaités
- Intérêts :
  1. Phase de recherche optimisée par rapport à MP
  2. Aussi facile à implémenter

Conclusion sur l'algorithme KMP (1977)

- Complexité : idem MP pour le pire des cas
  1. Pré-traitement : $O(m)$ en temps et en espace
  2. Phase de recherche : $O(n+m)$ en temps et en espace
- Une fois le pré-traitement effectué sur un motif, on peut le rechercher dans autant de textes que souhaités
- Intérêts :
  1. Phase de recherche optimisée par rapport à MP
  2. Aussi facile à implémenter

Peut-on encore faire mieux ?

Conclusion sur l'algorithme KMP (1977)

- Complexité : idem MP pour le pire des cas
  1. Pré-traitement : $O(m)$ en temps et en espace
  2. Phase de recherche : $O(n+m)$ en temps et en espace
- Une fois le pré-traitement effectué sur un motif, on peut le rechercher dans autant de textes que souhaités
- Intérêts :
  1. Phase de recherche optimisée par rapport à MP
  2. Aussi facile à implémenter

Peut-on encore faire mieux ? OUI !

Conclusion sur l'algorithme KMP (1977)

- Complexité : idem MP pour le pire des cas

  1. Pré-traitement : $O(m)$ en temps et en espace

  2. Phase de recherche : $O(n+m)$ en temps et en espace

- Une fois le pré-traitement effectué sur un motif, on peut le rechercher dans autant de textes que souhaités

- Intérêts :

  1. Phase de recherche optimisée par rapport à MP

  2. Aussi facile à implémenter

Peut-on encore faire mieux ? OUI !

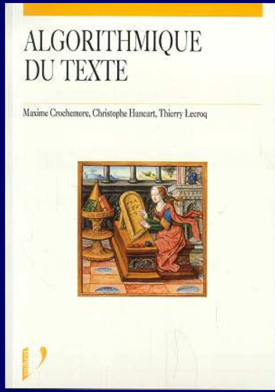
$\longrightarrow$ Recherche de droite à gauche (*cf.* Boyer-Moore)

- Introduction
- Algorithme naïf
- Algorithme MP
- Algorithme KMP
- Références

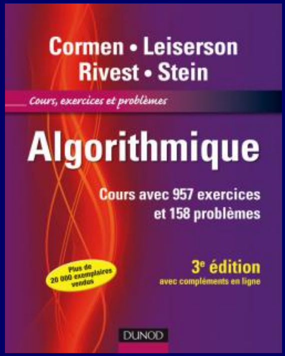
Références

- Ce cours a été construit à partir du cours de Maxime CROCHEMORE et Thierry LECROQ : Text searching

- Pour aller plus loin :



1. M. Crochemore, C. Hancart et T. Lecroq, Algorithmique du texte, Vuibert, 2001, 347 pages. ISBN 2-7117-8628-5. Disponible en version pdf : http://www-igm.univ-mlv.fr/~mac/CHL/CHL-2011.pdf



2. T. Cormen, C. Leiserson, R. Rivest et C. Stein, Algorithmique, Dunod, 2010 - 3e édition, 1296 pages. ISBN : 9782100545261
   ~ 15 exemplaires disponibles à la BU (2e et 3e édition)