

Recherche exacte de motif

Algotihmes naïf, MP et KMP

Sèverine Bérard

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

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

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

- Soit u un mot non vide, p un entier tel que $0 < p \leq |u|$
- p est une **période** de u si une de ces conditions est satisfaite :
 1. $u[i] = u[i + p]$, pour $1 \leq i \leq |u| - p$

- Soit u un mot non vide, p un entier tel que $0 < p \leq |u|$
- p est une **période** de u si une de ces conditions est satisfaite :
 1. $u[i] = u[i + p]$, pour $1 \leq i \leq |u| - p$
 2. u est un préfixe d'un mot y^k , $k > 0$, $|y| = p$

- Soit u un mot non vide, p un entier tel que $0 < p \leq |u|$
- p est une **période** de u si une de ces conditions est satisfaite :
 1. $u[i] = u[i + p]$, pour $1 \leq i \leq |u| - p$
 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

- Soit u un mot non vide, p un entier tel que $0 < p \leq |u|$
- p est une **période** de u si une de ces conditions est satisfaite :
 1. $u[i] = u[i + p]$, pour $1 \leq i \leq |u| - p$
 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 $|u|$)

- Soit u un mot non vide, p un entier tel que $0 < p \leq |u|$
- p est une **période** de u si une de ces conditions est satisfaite :
 1. $u[i] = u[i + p]$, pour $1 \leq i \leq |u| - p$
 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)

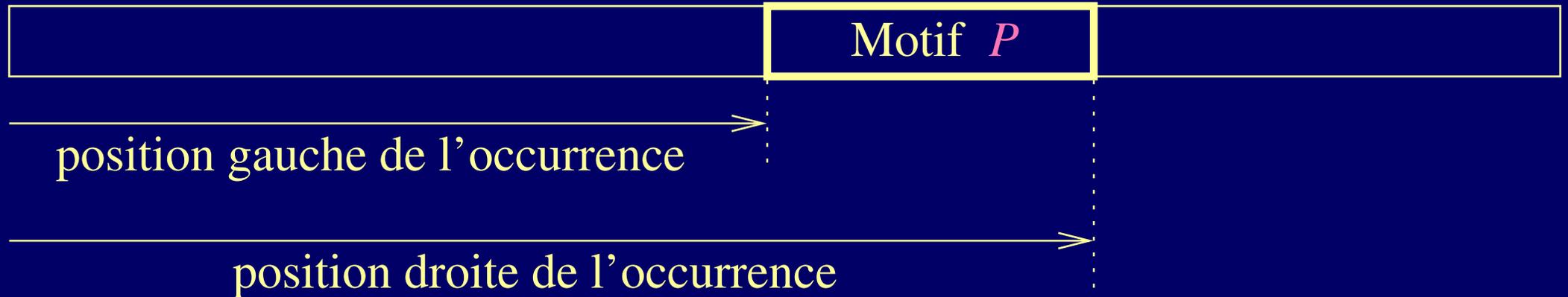
- Soit u un mot non vide, p un entier tel que $0 < p \leq |u|$
- p est une **période** de u si une de ces conditions est satisfaite :
 1. $u[i] = u[i + p]$, pour $1 \leq i \leq |u| - p$
 2. u est un préfixe d'un mot y^k , $k > 0$, $|y| = p$
 3. $u = yw = wz$, pour des mots y , z et w avec $|y| = |z| = p$
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	<i>abacaba</i>
8	<i>aba</i>
10	<i>a</i>
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

Texte T



- Motif : un mot P de longueur m

t a t a

- **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

- **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

- **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

- **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

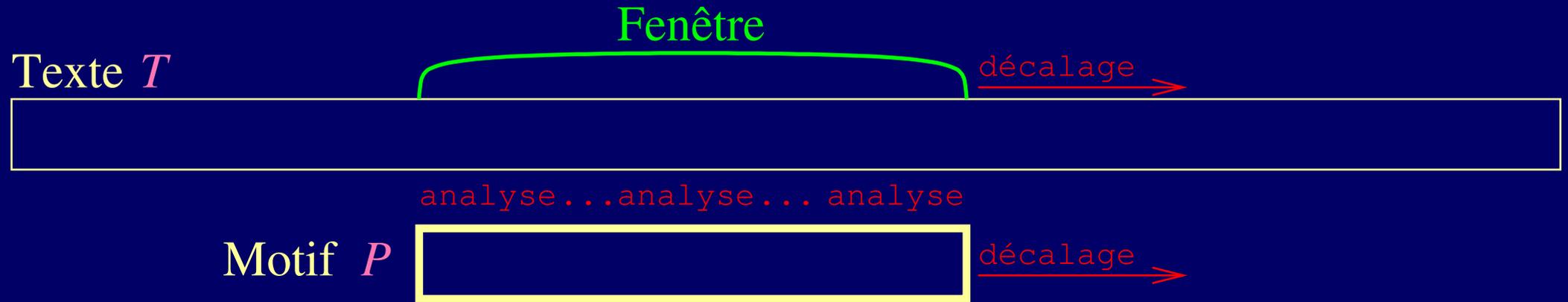
- **Occurrences** :

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 ($=, \neq$)





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

- 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 $pos \leq n - m + 1$ **faire**

$i := 1$;

tant que ($i \leq m$ **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



Motif P



Texte T



Motif P



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

Texte T



Motif P



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

Texte T



$border(u)$

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

Texte T



$border(u)$

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

Texte T



$border(u)$

- 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

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

- 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**

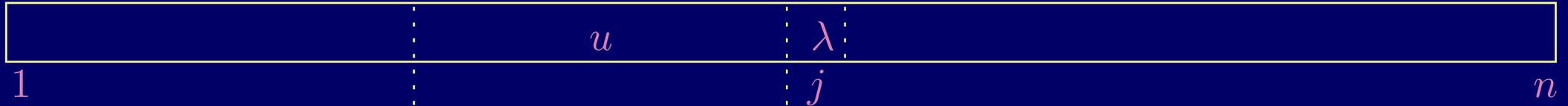
$u :=$ plus long préfixe commun entre la Fenêtre et le Motif ;

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

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

Mémoriser $border(u)$;

Texte T



Motif P



Algorithme 4: Algorithme MP

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

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

tant que $j \leq n$ **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 ", $j - i + 1$) ;

$i := MP_next[i] ;$

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

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

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

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

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

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

- i parcourt les longueurs des préfixes de manière croissante

- 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 ($j \geq 0$ **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

Remarque :

$$MP_next[i] =$$

Remarque :

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

Remarque :

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

⇒ Quasiment même algorithme

Remarque :

$MP_next[i] = BORD[i - 1] + 1$, pour i de 1 à $m + 1$

⇒ 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** $j := MP_next[j] ;$

$j := j + 1 ;$

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

Remarque :

$MP_next[i] = BORD[i - 1] + 1$, pour i de 1 à $m + 1$

⇒ 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 $m + 1$

⇒ 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

- Complexité :

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

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

- 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

- 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

- 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

- 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 ($O(m)$)

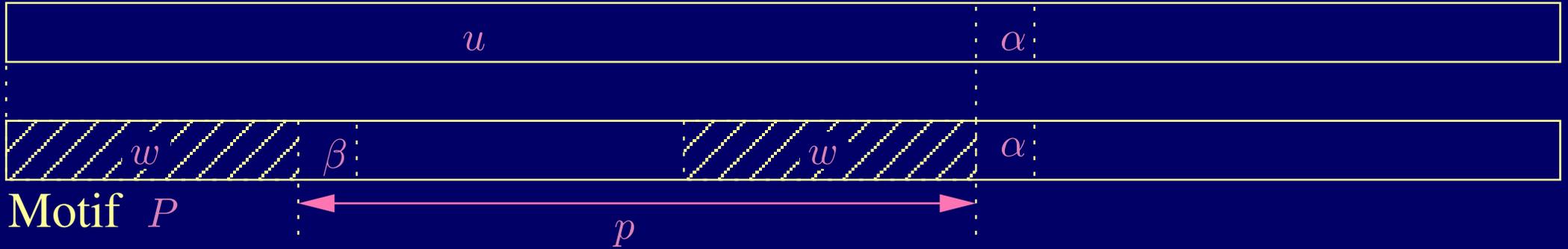
Peut-on faire mieux ?

- 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 ($O(m)$)

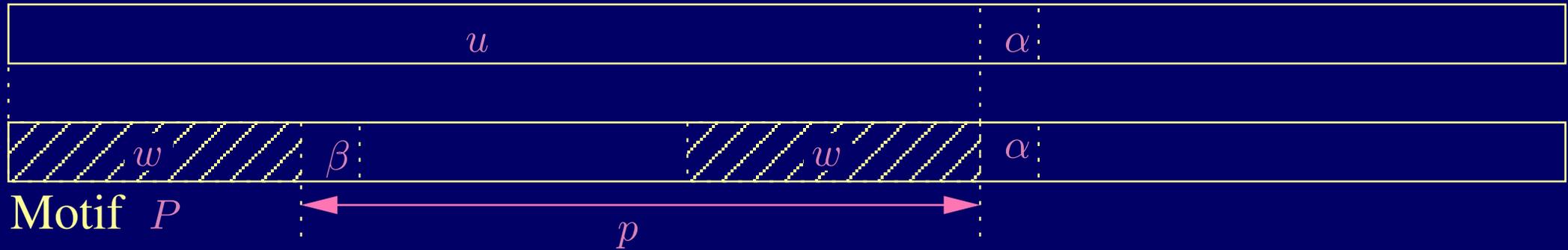
Peut-on faire mieux ? OUI!

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

Motif P

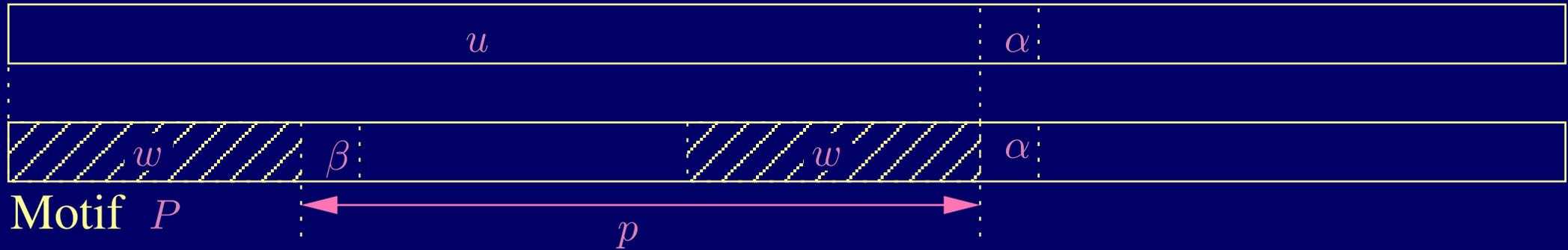


Motif P

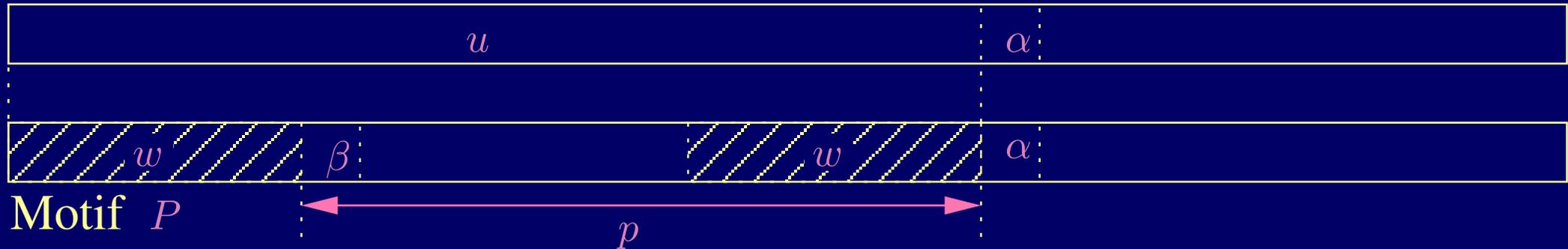


- Notions de bords stricts (w) et de périodes interrompues (p)

Motif P



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

Motif P 

- 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 $period_int(u)$ places vers la droite ;

Mémoriser $border_strict(u)$;

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

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

- 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

- 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 $u\beta$

- 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

- 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 $P = abacabacabacc$

- 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

- $k = MP_next[i]$

- $k = MP_next[i]$
- $KMP_next[i] = \begin{cases} k & \text{si } P[i] \neq P[k] \text{ ou si } i = m \\ KMP_next[k] & \text{si } P[i] = P[k] \end{cases}$

- $k = MP_next[i]$
- $KMP_next[i] = \begin{cases} k & \text{si } P[i] \neq P[k] \text{ ou si } i = m \\ KMP_next[k] & \text{si } P[i] = P[k] \end{cases}$

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

$KMP_next[i + 1] := KMP_next[j]$

- Complexité : idem MP pour le pire des cas

- Complexité : idem MP pour le pire des cas
 1. Pré-traitement : $O(m)$ en temps et en espace

- 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

- **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**

- **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

- **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

- **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 ?

- **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!**

- **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!**

→ Recherche de **droite à gauche** (cf. Boyer-Moore)

- Introduction
- Algorithme naïf
- Algorithme MP
- Algorithme KMP
- 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)

