

Recherche textuelle

- 1 Introduction – Position du problème
- 2 Recherche par force brute
- 3 L'algorithme de Boyer-Moore
 - L'algorithme de Boyer-Moore
 - L'algorithme de Boyer-Moore-Horspool
- 4 Complexité de BM – Autres algorithmes

- 1 Introduction – Position du problème
- 2 Recherche par force brute
- 3 L'algorithme de Boyer-Moore
 - L'algorithme de Boyer-Moore
 - L'algorithme de Boyer-Moore-Horspool
- 4 Complexité de BM – Autres algorithmes

Introduction – Position du problème

- En informatique les données textuelles sont présentes dans :
 - ▶ les grandes bases de données textuelles ;
 - ▶ les pages web ;
 - ▶ le code source des programmes ;
 - ▶ toutes les données encodées dans un alphabet.
- La recherche textuelle intervient entre autre :
 - ▶ dans le traitement automatique des langues (champ de l'Intelligence Artificielle), pour (par exemple) la traduction automatique ;
 - ▶ dans les algorithmes utilisés pour indexer avec des robots les (milliards de) site-web actifs ;
 - ▶ dans les navigateurs qui basent leur recherche sur des mots-clefs ;

- ▶ en bioinformatique pour le séquençage de l'ADN : on cherche une séquence donnée de nucléotides symbolisés par les lettres A, C, G et T dans une chaîne dont la longueur (pour l'homme) est de plusieurs milliards ;
- ▶ dans toutes les commandes (liées à une application, comme un éditeur de texte ou intégrées au système d'exploitation, comme la commande `find` sous linux) permettant à l'utilisateur de chercher une chaîne de caractère dans une autre.
- ▶ dans les fonctions liées au traitement des chaînes de caractères que proposent certains langages de programmation (méthode `find()` d'une variable de type `str` en Python).

- La recherche textuelle a comme objectif de déterminer dans un texte donné les occurrences d'un motif donné.
- Par exemple, avec le texte ci-dessous et le motif "miroir", on souhaite disposer d'une fonction `recherche(texte, motif)` qui renvoie le tableau des indices éventuels des occurrences du motif dans le texte, à savoir ici `[0, 17]`.

0	5	10	15	17	20	25
m i r o i r , m o n b e a u m i r o i r , d i s						
30	35	40	45	50	55	
- m o i q u i e s t l a p l u s b e l l e .						

- 1 Introduction – Position du problème
- 2 Recherche par force brute
- 3 L'algorithme de Boyer-Moore
 - L'algorithme de Boyer-Moore
 - L'algorithme de Boyer-Moore-Horspool
- 4 Complexité de BM – Autres algorithmes


Recherche par force brute

- Le premier algorithme envisageable est celui qu'on appelle «naïf» ou par «Force-Brute» (FB).
- Il consiste à déplacer le motif le long du texte, de gauche à droite, sur toutes les positions possibles du motif dans le texte.
- Pour chaque position du motif, on compare chaque lettre du motif avec celle du texte qui lui fait face, de gauche à droite en partant de la première lettre du motif, ceci tant qu'on a pas atteint la fin du motif et que les lettres sont égales.
- Si une des lettres du motif est différente de la lettre correspondante du texte, on décale le motif d'une lettre vers la droite et on recommence la comparaison du motif et du texte.
- Si toutes les lettres du motif sont égales à celles du texte, on mémorise la position du début du motif dans le texte, puis on décale le motif d'une lettre vers la droite et on recommence la comparaison du motif et du textes.

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b


a b a b b

1. Passer en plein écran pour visualiser cette diapositive. 

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b


a b a b b
 ↑

1. Passer en plein écran pour visualiser cette diapositive. 

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b


a	b	a	b	b
	↑			

1. Passer en plein écran pour visualiser cette diapositive. 

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b


a	b	a	b	b
		↑		

1. Passer en plein écran pour visualiser cette diapositive. 

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b


a	b	a	b	b
↑				

1. Passer en plein écran pour visualiser cette diapositive. 

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b


a	b	a	b	b
↑				

1. Passer en plein écran pour visualiser cette diapositive. 

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

		a	b	a	b	b
		↑				

1. Passer en plein écran pour visualiser cette diapositive. 

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹


0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

		a	b	a	b	b											
			↑														

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

		a	b	a	b	b											
						↑											

1. Passer en plein écran pour visualiser cette diapositive. 


Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

			a	b	a	b	b
			↑				

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹


0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b
					a	b	a	b	b								
					↑												

1. Passer en plein écran pour visualiser cette diapositive. 

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b


			a	b	a	b	b
				↑			

1. Passer en plein écran pour visualiser cette diapositive. 

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b


						a	b	a	b	b							
						↑											

1. Passer en plein écran pour visualiser cette diapositive. 

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹


0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

				a	b	a	b	b									
					↑												

1. Passer en plein écran pour visualiser cette diapositive. 

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹


0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b
							a	b	a	b	b						
								↑									

1. Passer en plein écran pour visualiser cette diapositive. 

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

				a	b	a	b	b									
									↑								

1. Passer en plein écran pour visualiser cette diapositive. 

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

						a	b	a	b	b							
										↑							

1. Passer en plein écran pour visualiser cette diapositive.

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b


						a	b	a	b	b
						↑				

1. Passer en plein écran pour visualiser cette diapositive. < □ > < ☰ > < ≡ > ≡ ↺ 🔍 ↻

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b


a	b	a	b	b
↑				

1. Passer en plein écran pour visualiser cette diapositive. 

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b


						a	b	a	b	b
						↑				

1. Passer en plein écran pour visualiser cette diapositive. 

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b


						a	b	a	b	b							
										↑							

1. Passer en plein écran pour visualiser cette diapositive. 

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

a b a b b
↑

1. Passer en plein écran pour visualiser cette diapositive. 

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b


a b a b b
↑

1. Passer en plein écran pour visualiser cette diapositive. < □ > < ☰ > < ≡ > ≡ ↺ 🔍 ↻

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

											a	b	a	b	b		
											↑						

1. Passer en plein écran pour visualiser cette diapositive. 

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

											a	b	a	b	b		
													↑				

1. Passer en plein écran pour visualiser cette diapositive. < □ > < ☰ > < ≡ > < ≡ > ≡ ↺ 🔍 ↻

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b


											a	b	a	b	b		
														↑			

1. Passer en plein écran pour visualiser cette diapositive. < □ > < ☰ > < ≡ > < ≡ > ≡ ↺ 🔍 ↻

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

											a	b	a	b	b		
																	↑

1. Passer en plein écran pour visualiser cette diapositive. 

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

												a	b	a	b	b
												↑				

1. Passer en plein écran pour visualiser cette diapositive. < □ > < ☰ > < ≡ > < ≡ > ≡ ↺ 🔍 ↻

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b


a b a b b
↑

1. Passer en plein écran pour visualiser cette diapositive. < □ > < 📄 > < ≡ > < ≡ > ≡ ↺ 🔍 ↻

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

													a	b	a	b	b
														↑			

1. Passer en plein écran pour visualiser cette diapositive. 

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme par FB¹

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

a b a b b
↑

1. Passer en plein écran pour visualiser cette diapositive. < □ > < ☰ > < ≡ > ≡ ↺ 🔍 ↻

L'algorithme, écrit en pseudo-code, qui met en œuvre les idées précédentes est donné ci-dessous :

Fonction RECHERCHE_FB(texte, motif)

▷ *Calcule et renvoie le tableau des éventuelles occurrences de la chaîne motif dans la chaîne texte. Algorithme par Force Brute.* ◀

$n \leftarrow \text{longueur}(\text{texte})$

$m \leftarrow \text{longueur}(\text{motif})$

initialiser positions avec un tableau vide

pour i allant de 0 à $n - m$ **faire**

$j \leftarrow 0$

tant que $j < m$ et $\text{texte}(i + j) = \text{motif}(j)$ **faire**

$j \leftarrow j + 1$

si $j = m$ **alors**

 ajouter i à positions

renvoyer positions



Ex. 1 Écrire en Python une fonction `recherche_FB(texte, motif)` qui implémente l'algorithme précédent. Tester la fonction sur plusieurs textes et motifs.

Rappels : en Python une chaîne de caractères est de type `str` et c'est un «iterable» : ses caractères sont repérés par un indice, le premier étant à l'indice 0, la longueur de la chaîne étant donnée par la fonction `len()`. Par exemple, si `motif` est la chaîne `"miroir"`, `motif[0]` est le caractère `'m'` et `len(miroir)` vaut 6.

- L'algorithme de recherche par FB n'offre malheureusement pas une garantie de complexité assurant de pouvoir l'utiliser dans tous les cas pour des textes volumineux.
- Dans le pire des cas, pour un motif de longueur m et un texte de longueur n il faut :
 - ▶ pour chaque position du motif réaliser m comparaisons ;
 - ▶ répéter ces comparaisons pour $n - m + 1$ positions possibles du motif.
- Le coût est donc de l'ordre de $m \times (n - m + 1)$, avec $0 \leq m \leq n$. On montre², que ce coût est maximum pour $m = \frac{n+1}{2}$ et qu'il vaut alors $\frac{(n+1)^2}{4}$.
- On voit que l'ordre de grandeur de la complexité du coût de la recherche par FB est au pire en n^2 pour un texte de taille n . Par exemple, pour la recherche d'un motif dans un texte de taille 1000, il faut déjà (au pire) de l'ordre d'un million de comparaisons pour effectuer la recherche.

2. petit exercice de recherche d'un maximum d'une fonction du second degré, entraînez-vous !

- Attention : cette complexité dans le pire des cas ne signifie pas que cet algorithme se comporte toujours mal. Au contraire, on montre que dans certains contextes (par exemple les chaînes de caractères issues des mots des langues naturelles), il a une complexité non pas quadratique mais linéaire en n .
- Cela vient de ce que dans la majorité des cas, la comparaison des lettres du motif et du texte échoue dès les premières lettres du motif.
- Par exemple, en Java, la méthode `indexOf()` de la classe `String`, utilise un algorithme par FB pour chercher une sous-chaîne à l'intérieur d'une autre.

- 1 Introduction – Position du problème
- 2 Recherche par force brute
- 3 L'algorithme de Boyer-Moore
 - L'algorithme de Boyer-Moore
 - L'algorithme de Boyer-Moore-Horspool
- 4 Complexité de BM – Autres algorithmes

L'algorithme de Boyer-Moore

- Un examen attentif de l'algorithme par force brute montre que certaines comparaisons de lettres peuvent être évitées.
- Exemple : page 7, lors de l'échec de la première comparaison du motif et du texte, sachant que la lettre du texte (un "c") n'est pas dans le motif, il est inutile de décaler le motif d'une seule lettre. On peut directement décaler le motif de trois lettres, ce qui accélère la recherche.
- L'originalité de l'algorithme de Boyer et Moore (BM) est de procéder aux comparaisons des lettres *en partant de la fin du motif*.

- Comme pour l'algorithme par FB, on compare le motif au texte en décalant le motif de gauche à droite par rapport au texte.
- À chaque position du motif, on commence la comparaison en commençant par les dernières lettres du motif.
- Quand une des lettres du texte est différente de la lettre correspondante du motif :
 - 1 si cette lettre ne figure pas dans la partie du motif qui n'a pas encore été testée, on décale le motif de sorte que sa première lettre soit située *après* la lettre du texte, et on recommence la comparaison du texte et du motif ;
 - 2 si cette lettre figure (au moins une fois) dans la partie du motif qui n'a pas encore été testée, on décale celui-ci de sorte à faire coïncider cette lettre (la première rencontrée en «remontant» le motif) avec celle du texte, puis on recommence la comparaison du texte et du motif.
- Si toutes les lettres du texte et du motif sont égales, on mémorise la position du motif dans le texte, on décale le motif d'une lettre et on recommence la comparaison du texte et du motif.

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme de Boyer et Moore³

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

a b a b b

3. Passer en plein écran pour visualiser cette diapositive.

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme de Boyer et Moore³

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

a	b	a	b	b
			↑	

3. Passer en plein écran pour visualiser cette diapositive.

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme de Boyer et Moore³

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

a	b	a	b	b
			↑	

3. Passer en plein écran pour visualiser cette diapositive.

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme de Boyer et Moore³

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

a	b	a	b	b
				↑

3. Passer en plein écran pour visualiser cette diapositive.

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme de Boyer et Moore³

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

		a	b	a	b	b
						↑

3. Passer en plein écran pour visualiser cette diapositive.

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme de Boyer et Moore³

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

		a	b	a	b	b											
						↑											

3. Passer en plein écran pour visualiser cette diapositive.

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme de Boyer et Moore³

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

a b a b b

↑

3. Passer en plein écran pour visualiser cette diapositive.

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme de Boyer et Moore³

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

a b a b b

↑

3. Passer en plein écran pour visualiser cette diapositive.

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme de Boyer et Moore³

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

a b a b b

 ↑


3. Passer en plein écran pour visualiser cette diapositive.

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme de Boyer et Moore³

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

a b a b b

↑

3. Passer en plein écran pour visualiser cette diapositive. 

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme de Boyer et Moore³

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

a b a b b
↑

3. Passer en plein écran pour visualiser cette diapositive.

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme de Boyer et Moore³

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

a b a b b

↑

3. Passer en plein écran pour visualiser cette diapositive.

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme de Boyer et Moore³

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b
											a	b	a	b	b		
																	↑

3. Passer en plein écran pour visualiser cette diapositive.

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme de Boyer et Moore³

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b
													a	b	a	b	b
																	↑

3. Passer en plein écran pour visualiser cette diapositive.

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme de Boyer et Moore³

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

													a	b	a	b	b
																	↑

3. Passer en plein écran pour visualiser cette diapositive.

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme de Boyer et Moore³

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b	
														a	b	a	b	b
																		↑

3. Passer en plein écran pour visualiser cette diapositive.

Exemple : on recherche le motif "a b a b b" dans le texte ci-dessous par l'algorithme de Boyer et Moore³

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
a	b	c	a	b	a	a	b	a	b	b	a	b	a	b	a	b	b

													a	b	a	b	b
													↑				

3. Passer en plein écran pour visualiser cette diapositive.

Ex. 2 Soit le texte "abcabac**baba**abcab" et le motif "abaabc". Écrire sur une feuille quadrillée le texte puis, en appliquant l'algorithme de recherche textuelle de BM, écrire sous le texte les différentes positions que prend le motif, ligne sous ligne.

Ex. 3 On suppose que le texte est cette fois-ci la chaîne de nucléotides "ATAACAGGAGTAAATAACGGCTGGAGTA" et que le motif recherché est "CGGCTG". Réaliser, sur une feuille quadrillée, le même traitement que celui demandé à l'exercice précédent.

- Le principe de l'algorithme de BM montre qu'il est possible (et même souhaitable, pour des raisons d'efficacité) de précalculer les décalages à appliquer au motif dans le cas 2 (et seulement dans ce cas, le précédent ne dépendant pas du caractère du texte trouvé mais seulement de l'indice du caractère du motif).
- Chacun de ces décalages dépend :
 - ▶ de l'indice j de la lettre du motif couramment testée (avec $1 \leq j \leq m - 1$, où m est la longueur du motif) ;
 - ▶ du caractère c du texte qui lui fait face et qui figure dans la partie du motif précédant strictement la lettre d'indice j .
- La structure de données à utiliser pour les décalages est donc à deux entrées. On peut remarquer que le nombre de lettres qui précède la lettre d'indice j est variable (et dépend du motif). Il est donc naturel, en Python, de prendre pour chaque j un dictionnaire ayant autant de clefs qu'il y a de caractères avant celui d'indice j dans le motif.
- On donne page suivante deux exemples de tableaux de décalages associés à un motif, et (colonne de droite) leur construction en Python sous forme d'un tableau de dictionnaires.

Exemple 1

motif	0	1	2	3	4	5
	a	b	c	a	b	c

	a	b	c
1	1		
2	2	1	
3	3	2	1
4	1	3	2
5	2	1	3

```
d = [{"a": 1},
      {"a": 2, "b": 1},
      {"a": 3, "b": 2, "c": 1},
      {"a": 1, "b": 3, "c": 2},
      {"a": 2, "b": 1, "c": 3}]
```

Explications :

Exemple 1

motif	0	1	2	3	4	5
	a	b	c	a	b	c

	a	b	c
1	1		
2	2	1	
3	3	2	1
4	1	3	2
5	2	1	3

```
d = [{"a": 1},
      {"a": 2, "b": 1},
      {"a": 3, "b": 2, "c": 1},
      {"a": 1, "b": 3, "c": 2},
      {"a": 2, "b": 1, "c": 3}]
```

Explications :

Pour $j = 2$, la partie restant à tester du motif est "ab". Elle contient la lettre "a", située à une distance de 2 lettres et la lettre b située à 1 lettre (de la lettre "c" d'indice 2).

Exemple 1

	0	1	2	3	4	5
motif	a	b	c	a	b	c

	a	b	c
1	1		
2	2	1	
3	3	2	1
4	1	3	2
5	2	1	3

```

d = [{"a": 1},
     {"a": 2, "b": 1},
     {"a": 3, "b": 2, "c": 1},
     {"a": 1, "b": 3, "c": 2},
     {"a": 2, "b": 1, "c": 3}]
  
```

Explications :

Pour $j = 3$, la partie restant à tester du motif est "abc". Elle contient la lettre "a", située à une distance de 3 lettres, la lettre b située à 2 lettres et la lettre "c" située à une distance de 1 lettre (de la lettre "a" d'indice 3).

Exemple 2

motif	0	1	2	3	4	5	6
	a	b	c	c	a	b	c

	a	b	c
1	1		
2	2	1	
3	3	2	1
4	4	3	1
5	1	4	2
6	2	1	3

```
d = [{"a": 1},
      {"a": 2, "b": 1},
      {"a": 3, "b": 2, "c": 1},
      {"a": 4, "b": 3, "c": 1},
      {"a": 1, "b": 4, "c": 2},
      {"a": 2, "b": 1, "c": 3}]
```

Explications :

Exemple 2

motif	0	1	2	3	4	5	6
	a	b	c	c	a	b	c

	a	b	c
1	1		
2	2	1	
3	3	2	1
4	4	3	1
5	1	4	2
6	2	1	3

```
d = [{"a": 1},
      {"a": 2, "b": 1},
      {"a": 3, "b": 2, "c": 1},
      {"a": 4, "b": 3, "c": 1},
      {"a": 1, "b": 4, "c": 2},
      {"a": 2, "b": 1, "c": 3}]
```

Explications :

Pour $j = 3$, la partie restant à tester du motif est "abc". Elle contient la lettre "a", située à une distance de 3 lettres, la lettre b située à 2 lettres et "c" située à 1 lettre (du "c" d'indice 3).

Exemple 2

motif	0	1	2	3	4	5	6
	a	b	c	c	a	b	c

	a	b	c
1	1		
2	2	1	
3	3	2	1
4	4	3	1
5	1	4	2
6	2	1	3

```
d = [{"a": 1},
      {"a": 2, "b": 1},
      {"a": 3, "b": 2, "c": 1},
      {"a": 4, "b": 3, "c": 1},
      {"a": 1, "b": 4, "c": 2},
      {"a": 2, "b": 1, "c": 3}]
```

Explications :

Pour $j = 5$, la partie restant à tester du motif est "abcca". Elle contient la lettre "a", située à une distance de 1 lettre, la lettre b située à 4 lettres et la lettre "c" située à une distance de 2 lettres (de la lettre "b" d'indice 5).

Ex. 4 Construire le tableau à deux entrées des décalages à précalculer dans l'algorithme de recherche textuelle de BM pour les motifs suivants : "banane" et "chercher".

Remarques :

- Il est intéressant de rajouter une ligne contenant un dictionnaire vide au début du tableau précédent. D'une part cela permet de faire en sorte que sa j^{e} ligne contienne bien le dictionnaire des décalages associés au traitement de la lettre d'indice j (sinon il faudrait tenir compte d'un décalage de 1). D'autre part on peut de cette façon traiter identiquement le cas $j = 0$ et le cas où la lettre du texte n'est pas une lettre du motif, ces deux cas étant obtenu par le test de l'appartenance de la lettre du texte comme étant une clef du j^{e} dictionnaire du tableau. En effet, dans ces deux cas le décalage à appliquer est de $j + 1$ lettres.
- On peut remarquer que la distance de la j^{e} lettre du motif à elle-même n'a pas à être calculée (dans le cas où cette j^{e} lettre a une autre occurrence avant celle d'indice j), ce qui fait que certaines valeurs du tableau ne seront en fait jamais utilisées. En effet, par construction, le décalage n'est réalisé que si la lettre du texte est distincte de cette j^{e} lettre du motif. Dans l'exemple 2 précédent, $d[5][\text{motif}[5]] = d[5][\text{"b"}]$ ne sera jamais utilisé.

- Pour construire les distances du dictionnaire de rang j ($j \geq 1$) on remarque qu'il suffit de parcourir de gauche à droite les j lettres du motif qui précèdent la j^e à l'aide d'un indice k ($0 \leq k \leq j - 1$). La valeur du dictionnaire associée à chacune des lettres rencontrées sera simplement la dernière valeur $j - k$ associée à la dernière occurrence de cette lettre.
- La synthèse des idées précédentes est résumée dans l'algorithme ci-dessous écrit en Python.

```
def decalages_BM(motif):
    d = [{} for _ in range(len(motif))]
    for j in range(1, len(motif)):
        for k in range(j):
            d[j][motif[k]] = j - k
        d[j].pop(motif[j], 0) # suppression d'une (éventuelle)
                             # clé inutile
    return d
```

- On donne page suivante l'algorithme de Boyer-Moore de recherche d'un motif dans un texte.

Fonction RECHERCHE_BM(texte, motif)

▷ *Revoie la liste des occurrences de la chaîne motif dans la chaîne texte.
Algorithme de Boyer-Moore.* ◀

$n \leftarrow \text{longueur}(\text{texte})$

$m \leftarrow \text{longueur}(\text{motif})$

$d \leftarrow \text{DECALAGES_BM}(\text{motif})$

initialiser positions avec un tableau vide

$i \leftarrow 0$

tant que $i \leq n - m$ **faire**

$j \leftarrow m - 1$

tant que $j \geq 0$ et $\text{texte}(i + j) = \text{motif}(j)$ **faire**

$j \leftarrow j - 1$

si $j = -1$ **alors**

 ajouter i à positions

$\text{decal} \leftarrow 1$

sinon

si $\text{texte}(i + j)$ n'est pas une clef de $d(j)$ **alors**


$\text{decal} \leftarrow j + 1$


sinon

$\text{decal} \leftarrow d(j, \text{texte}(i + j))$

$i \leftarrow i + \text{decal}$

renvoyer positions

 Ex. 5 En utilisant l'algorithme précédent, écrire en Python le code d'une fonction `recherche_BM(texte, motif)` qui calcule et renvoie le tableau des indices des éventuelles occurrences de la chaîne `motif` dans la chaîne `texte` à l'aide de l'algorithme de Boyer-Moore. Cette fonction utilisera la fonction `decalages_BM(motif)` donnée page 21. Tester cette fonction sur un jeu d'essais suffisant.

 Ex. 6 On souhaite comparer les temps d'exécution des recherches textuelles réalisées avec l'algorithme de BM et avec l'algorithme par FB. Pour cela, on se propose de générer un texte constitué d'une suite aléatoire d'un million de caractères choisis parmi les quatre lettres "A", "T", "C" et "G" et motif contenant 1000 caractères choisis dans le même ensemble (tirés eux aussi aléatoirement). Réaliser un programme qui, effectue la recherche du motif dans le texte (tous deux générés aléatoirement) et qui affiche les temps de recherche des deux algorithmes (indications page suivante).

Rappels : pour mesurer un temps d'exécution, on peut utiliser le module `time` comme indiqué dans le bout de code ci-dessous

```
from time import *

debut = perf_counter()
# bloc dont on veut
# mesurer le temps
# d'exécution
fin = perf_counter()
temps = fin - debut # temps d'exécution
```

Pour générer une chaîne de longueur donnée composée uniquement des caractères "ACGT" on peut procéder comme indiqué ci-dessous :

```
from random import randint

choix = "ACGT"
texte = [choix[randint(0, 3)] for i in range(10000)]
```

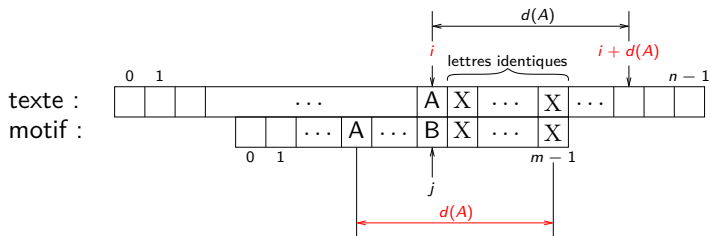
L'algorithme de Boyer-Moore-Horspool

- L'algorithme de Boyer-Moore-Horspool (BMH) est une version simplifiée (et donc un peu moins performante) de l'algorithme de BM qui utilise un tableau de décalages à une entrée (au lieu de deux). Chaque décalage est uniquement associé à la lettre rencontrée dans le texte différente de celle qui lui fait face dans le motif, pourvu que cette lettre figure dans le motif.
- Plus précisément, on calcule pour chaque lettre du motif la distance qui la sépare de la fin du motif. Dans le cas particulier de la dernière lettre du motif, on ne calcule cette distance que si cette dernière lettre figure aussi ailleurs dans (au moins) une autre position du motif.
- Par exemple, si le motif est "abcabc", le dictionnaire d utilisé en Python peut être initialisé par $d = \{"a": 2, "b": 1, "c": 3\}$. Pour le motif "ababc", $d = \{"a": 2, "b": 1\}$.

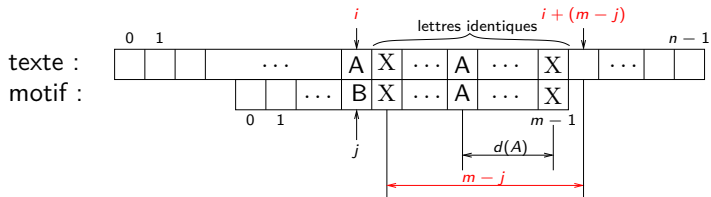
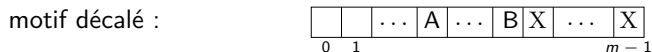
- L'hypothèse implicite de BMH est que la majorité des décalages du motif accélérant la recherche sont ceux où la lettre du texte qui ne correspond pas à celle du motif est rencontrée pour la première fois dans le motif. Dans ce cas, le décalage de BMH est le même que celui de BM.
- Les distances de l'algorithme de BMH n'étant pas calculées comme celles de BM, le calcul des décalages est également différent.
- On a de ce fait intérêt à utiliser, dans l'algorithme de BMH, un indice i (initialisé à une valeur comprise entre $m - 1$ et $n - 1$) pointant sur la lettre du texte couramment testée. Cet indice varie en même temps que l'indice j qui repère la lettre du motif couramment testée. Avec ce choix d'indice, le décalage de BMH consiste simplement (la plupart du temps) à ajouter la distance précalculée à i .

Il y a quatre cas de figures pour le décalage du motif. Ces cas sont explicités ci-dessous et illustrés page suivante par une figure. Pour les 3 premiers cas on note c la première lettre du texte, d'indice i , différente de la lettre du motif lui faisant face, d'indice j .

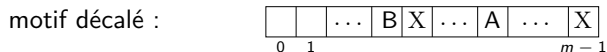
- 1 c n'a pas encore été rencontrée dans le motif. Dans ce cas, on décale le motif de $d(c)$ lettres, où $d(c)$ est la distance de BMH précalculée pour c . Cela revient à incrémenter i de $d(c)$.
- 2 c a déjà été rencontrée dans le motif. Dans ce cas, on décale le motif d'une seule lettre (et on n'utilise pas la distance précalculée, ce qui ferait revenir le motif en arrière!). Cela revient à incrémenter i de $m - j$.
- 3 c ne figure pas dans le motif. Dans ce cas on décale le motif de $j + 1$ lettres. Cela revient à incrémenter i de m .
- 4 Toutes les lettres du texte sont égales à celles du motif : on décale le motif d'une lettre. Cela revient à incrémenter i de $m + 1$.



Cas ①



Cas ②



- Les quatre cas de figure précédents conduisent à des décalages du motif qui se traduisent par une incrémentation de i . Selon le cas, l'incrément à ajouter est :
 - ① $d(c)$ si $d(c) \geq m - j$ (c étant la lettre du texte trouvée figurant dans le motif et $d(c)$ sa distance de BMH précalculée) ;
 - ② $(m - j)$, lorsque $d(c) \leq m - j - 2$
 - ③ $d(c) = m$, à condition d'associer une distance de m à toute lettre c qui ne figure pas dans le motif ;
 - ④ $m + 1$ lorsque $j = -1$ est atteint.
- On remarque que ces 4 incréments peuvent être calculés à l'aide de l'unique formule $\max(d(c), m - j)$ pourvu que le dictionnaire d des décalages intègre un décalage de m pour toute lettre du texte ne figurant pas dans le motif (cas ③).
- Toutes ces idées conduisent à l'algorithme de recherche de BMH donné page suivante en pseudo-code.

Fonction RECHERCHE_BMH(texte, motif)

▷ *Renvoie la liste des occurrences de la chaîne motif dans la chaîne texte.
Algorithme de Boyer-Moore-Horspool.* ◀

$n \leftarrow \text{longueur}(\text{texte})$

$m \leftarrow \text{longueur}(\text{motif})$

$d \leftarrow \text{DECALAGES_BMH}(\text{texte}, \text{motif})$

initialiser positions avec un tableau vide

$i \leftarrow m - 1$

tant que $i < n$ **faire**

$j \leftarrow m - 1$

tant que $j \geq 0$ et $\text{texte}(i) = \text{motif}(j)$ **faire**

$i \leftarrow i - 1$

$j \leftarrow j - 1$

si $j = -1$ **alors**

 ajouter $i + 1$ à positions

$i \leftarrow i + \max(d(\text{texte}(i), m - j)$

renvoyer positions



Ex. 7 Traduire l'algorithme précédent en Python en créant une fonction `recherche_BMH(texte, motif)` qui calcule et renvoie le tableau des occurrences éventuelles de la chaîne `motif` dans la chaîne `texte`. On codera provisoirement «en dur» le dictionnaire `d` des distances de BMH pour un motif donné (la fonction `decalages_BMH(texte, motif)` est construite dans l'exercice suivant). Tester l'algorithme sur plusieurs textes/motifs.

Une possibilité, pour écrire l'algorithme qui calcule le dictionnaire `d` des décalages de BMH associés au motif et au texte est d'utiliser l'encodage avec lequel les chaînes `texte` et `motif` ont été saisies. De cette façon il n'est pas nécessaire de parcourir tous les caractères de la chaîne `texte` pour les découvrir (ce qui peut être pénalisant si elle fait plusieurs milliards de caractères de long). On initialise d'abord `d` avec tous les caractères de cet encodage auxquels on affecte une distance de `m`. Puis on parcourt tous les caractères de la chaîne `motif` de gauche à droite. Pour chaque caractère `c` d'indice `i` dans `motif`, sa distance à la dernière lettre est la dernière valeur `m-i-1` qui sera affectée à `d[c]` (car `m-i-1` est le nombre de lettres du motif entre celle d'indice `i+1` et la dernière d'indice `m-1`).



Ex. 8 Écrire en Python une fonction `decalages_BMH(texte, motif)` qui calcule et renvoie le dictionnaire des distances de BMH des lettres de la chaîne `texte` par rapport à celles de la chaîne `motif`. On pourra par exemple supposer que l'encodage utilisé est le code ASCII étendu contenant 256 caractères et permettant de coder (entre autre) les lettres accentuées des alphabets latins occidentaux. Tester ensuite cette fonction en l'appelant depuis la fonction `recherche_BMH`.

Rappel : en Python, les 256 caractères du code ASCII étendu sont ceux de la forme `chr(i)`, avec `i` variant de 0 à 255.

- 1 Introduction – Position du problème
- 2 Recherche par force brute
- 3 L'algorithme de Boyer-Moore
 - L'algorithme de Boyer-Moore
 - L'algorithme de Boyer-Moore-Horspool
- 4 Complexité de BM – Autres algorithmes

Complexité de BM – Autres algorithmes

- L'étude fine de la complexité de l'algorithme de BM est difficile et hors-programme.
- On peut démontrer que, dans des situations typiques, la complexité de BM est en $\frac{n}{m}$, c'est à dire «sous-linéaire».
- Pour le voir intuitivement : il suffit que le motif ne contienne qu'une petite fraction des lettres de l'alphabet avec lequel est composé le texte. Dans ce cas, la majorité des comparaisons conduisent à un décalage de m lettres du motif, d'où un nombre moyen de $\frac{n}{m}$ comparaisons.
- Pour finir indiquons l'existence d'autres algorithmes de recherche textuelle : Morris-Pratt-Knuth ou Rabin-Karp (parmi les plus connus). Le premier a une version dont la complexité est garantie en $2n$, le dernier en $7n$. Ces algorithmes ont une histoire récente (années 1970-1980).

Il est intéressant de noter que l'algorithme de Morris-Knuth-Pratt a été suscité par un résultat d'informatique théorique (obtenu en 1970) de S. Cook⁴, prouvant l'existence d'un algorithme de recherche textuelle de complexité en $n + m$, sans pour autant explicitement donner celui-ci. D. Knutt et V. Pratt ont réussi à exhiber un tel algorithme en s'inspirant des travaux de Cook. Au même moment et indépendamment, un autre informaticien, J. Morris, a également découvert ce même algorithme. On voit là un exemple d'un phénomène typique en histoire des sciences, et cette anecdote tend à conforter l'idée que l'informatique a désormais acquis le statut d'une discipline scientifique à part entière, où théorie et pratique sont intimement mêlées.

4. S. Cook est un des fondateurs de la théorie de la complexité et de la théorie des problèmes NP-complets.