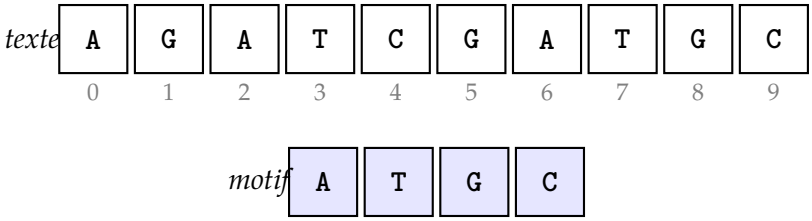
Recherche textuelle : activités de découverte

① Activité 1 : chercher un motif dans une séquence ADN

Remarque 1 (Contexte). En bioinformatique, on cherche souvent à repérer un motif court (quelques bases) dans une longue séquence ADN (plusieurs millions de bases). Dans cette activité, on se limite à une séquence de dix bases et à un motif de quatre bases, pour pouvoir dérouler la recherche à la main.
On note les quatre bases de l'ADN `A`, `C`, `G` et `T`. La séquence (le **texte**) et le **motif** recherché sont donnés ci-dessous.



Exercice 1 *Dérouler la recherche naïve* La stratégie la plus simple est la suivante : on aligne le motif tout à gauche du texte, on compare les caractères de **gauche à droite** ; dès qu'un caractère ne correspond pas, on fait glisser le motif d'**une seule** position vers la droite et on recommence.

1. Compléter le tableau ci-dessous en indiquant, pour chaque position i du motif, le nombre de comparaisons effectuées et le résultat (motif trouvé ou non).

Position i	Alignement	Nombre de comparaisons	Trouvé ?
0	`ATGC` aligné sur `AGAT`		
1			
2			
3			
4			
5			
6			

2. Donner la position du motif dans le texte, ou indiquer qu'il n'y est pas.

3. Quel est le nombre total de comparaisons effectuées ?

4. Le texte a pour longueur $n = 10$ et le motif $m = 4$. Combien de positions i l'algorithme a-t-il testées ? Exprimer ce nombre en fonction de n et m.

Exercice 2 *Comptage du pire cas* On considère maintenant le texte `AAAAAAAAAB` et le motif `AAAB`.

1. À quelle position le motif apparaît-il dans le texte ?

2. Pour chaque position i de 0 à $n - m$, combien de comparaisons effectue-t-on avant de détecter un échec (ou un succès final) ?

3. En déduire le nombre total de comparaisons.

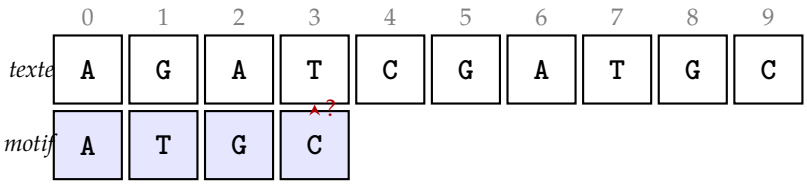
4. De façon générale, si un texte de longueur n et un motif de longueur m sont tous les deux composés presque uniquement d'une seule lettre, quel est l'ordre de grandeur du nombre de comparaisons ?

② Activité 2 : et si on comparait à l'envers ?

Remarque 2 (Idée clé). L'algorithme naïf avance toujours d'**une seule** position après un échec, même si cet échec apporte beaucoup d'information. En comparant le motif **de droite à gauche**, on va pouvoir réutiliser le caractère du texte qui a provoqué l'échec pour **sauter** plusieurs positions d'un coup.

On reprend le texte `AGATCGATGC` et le motif `ATGC`. On va aligner le motif sur le texte et comparer en commençant par la **dernière** lettre du motif.

Exercice 3 *Premier alignement : un caractère étranger* Position $i = 0$: le motif `ATGC` est aligné avec `AGAT`. On compare la **dernière** lettre du motif (`C`) avec la lettre du texte à la position $i + m - 1 = 3$.



1. La lettre comparée dans le texte est `T`. Est-elle présente dans le motif ?

2. Si l'on décale le motif d'une seule position vers la droite, la lettre `T` du texte se retrouve-t-elle alignée avec une lettre qui correspond ?

3. Si au contraire la lettre comparée avait été, par exemple, `X` (absente du motif), quel aurait été le décalage le plus grand possible qui **garantirait** de ne pas rater une occurrence du motif ?

Exercice 4 *Formaliser la règle de décalage*

1. Soit c le caractère du texte à la position $i + m - 1$ (celui aligné avec la dernière lettre du motif).

   Compléter les deux règles suivantes :

   Règle 1 — si c n'apparaît pas dans le motif : on décale le motif de ..... positions.

   Règle 2 — si c apparaît dans le motif à la position k (la plus à droite possible, mais strictement avant la dernière) : on décale le motif de ..... positions.

2. Justifier la formule de la règle 2 par un petit dessin (le c du motif situé à la position k doit venir s'aligner avec le c du texte situé à la position $i + m - 1$).

3. Pourquoi exclut-on la dernière position du motif dans la règle 2 ? (Astuce : que se passerait-il si c était la dernière lettre du motif elle-même ?)

③ Activité 3 : construire la table de décalages

Remarque 3 (Objectif). Pour éviter de refaire le raisonnement précédent à chaque échec, on pré-calcule une fois pour toutes, avant la recherche, un **dictionnaire** qui donne, pour chaque lettre, le décalage à appliquer si cette lettre provoque un échec à la fin du motif.
Règle rappelée : pour chaque caractère c du motif **sauf le dernier**, on cherche sa position la plus à droite k dans `motif`$[0 \dots m-2]$, et on pose `table`$[c] = m - 1 - k$. Les caractères qui n'apparaissent pas dans la table reçoivent implicitement le décalage m.

Exercice 5 *Cas simple : motif à lettres toutes distinctes* On considère le motif `NSI` (longueur $m = 3$).

1. Remplir le tableau suivant.

Caractère	Position(s) dans `motif`$[0 \dots m-2]$	Position la plus à droite k	Décalage $m - 1 - k$
`N`			
`S`			
`I`	(dernière lettre, exclue)	–	–

2. En déduire le dictionnaire `table` renvoyé par le pré-traitement.

3. Quel décalage appliquerait-on si le caractère du texte aligné avec la fin du motif était `Z` ? Pourquoi ?

Exercice 6 *Cas avec lettres répétées* On considère le motif `ANANA` (longueur $m = 5$).

1. Compléter le tableau.

Caractère	Positions dans `motif`$[0\ldots3]$	k (la plus à droite)	Décalage
`A`			
`N`			

2. En déduire `table`.

3. Pourquoi, pour `A`, garde-t-on la position 2 et non la position 0 ?

Exercice 7 *Application sur un motif plus long* Construire la table de décalages pour le motif `MAMAN` (longueur 5). Donner explicitement le dictionnaire obtenu.

④ Activité 4 : dérouler l'algorithme complet

Remarque 4 (Synthèse de l'algorithme). On dispose d'un texte de longueur n, d'un motif de longueur m, et on a pré-calculé la table des décalages. L'algorithme procède ainsi :

1. on initialise $i = 0$ (position de début de la fenêtre dans le texte) ;
2. tant que $i \leq n - m$:
   - on compare le motif au texte **de droite à gauche** ;
   - si tout correspond, on a trouvé une occurrence à la position i et on avance de 1 ;
   - sinon, on regarde le caractère c du texte en position $i + m - 1$: on ajoute `table`$[c]$ à i (ou m si c n'est pas dans la table).

Exercice 8 *Rechercher `NSI` dans un petit texte* On reprend le motif `NSI` (table `{'N': 2, 'S': 1}`) et on le cherche dans le texte `LYCEENSIETNSI` (longueur $n = 13$).

L	Y	C	E	E	N	S	I	E	T	N	S	I
0	1	2	3	4	5	6	7	8	9	10	11	12

Compléter le tableau ci-dessous. À chaque ligne : la position i en début d'itération, le caractère aligné avec la fin du motif (texte à la position $i + m - 1$), le résultat (motif trouvé ou non) et le décalage appliqué.

Étape	i	$i+m-1$	Caractère aligné c	Résultat	Décalage appliqué
1	0	2			
2					
3					
4					
5					
6					

1. Aux positions où le motif a été trouvé : .....
2. Nombre total de comparaisons effectuées par Boyer-Moore-Horspool : .....

Exercice 9 *Comparaison avec l'algorithme naïf*

1. Sur le même texte `LYCEENSIETNSI` et le même motif `NSI`, combien de comparaisons aurait effectuées l'algorithme naïf ? (Détailler par position.)

2. Comparer ce nombre avec celui obtenu par Boyer-Moore-Horspool. Combien de comparaisons a-t-on économisées ? Exprimer le gain en pourcentage.

3. Pour quel type de texte et de motif Boyer-Moore-Horspool est-il particulièrement efficace ? Dans quel cas ne gagne-t-il rien par rapport à l'algorithme naïf ?

Exercice 10 *Pour aller plus loin : recherche de `ATGC`* Reprendre l'exemple de l'activité 1 : texte `AGATCGATGC`, motif `ATGC`.

1. Construire la table de décalages du motif `ATGC`.

2. Dérouler l'algorithme de Boyer-Moore-Horspool sur ce texte.

3. Comparer le nombre de comparaisons avec celui obtenu à l'activité 1 (rappel : 13 comparaisons pour l'algorithme naïf).