II. Algorithme de Boyer-Moore

1) Accélération de la recherche naïve

La fonction écrite précédemment permet de faire une recherche accélérée au sens où la comparaison des chaines de caractères s'arrête dès la **première différence** entre le motif et le texte. Cette comparaison doit tout de même être faite pour chaque décalage possible s…

❐ — À faire vous-même **3** —

On s'intéresse au texte : $\mathcal{T} =$ `"bricabrac"` et au motif $\mathcal{M} =$ `"bra"`.

- ❖ Expliquez pourquoi, lorsque vous effectuez une recherche du motif $\mathcal{M}$, vous n'avez pas besoin de considérer les sous-chaines `"ric"` et `"ica"`.
- ❖ On a ici comparé 3 lettres (à savoir "b", "r" et "a") pour répondre à la première question. Quel changement simple pourrait-on faire pour gagner quelques comparaisons ?

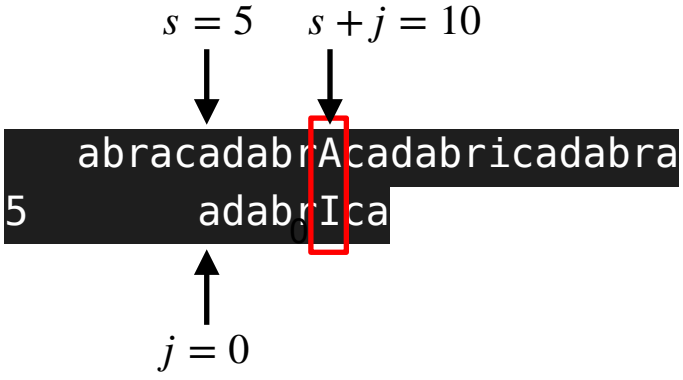
Cet exemple nous montre que l'on peut sauter certaines sous-chaines pour améliorer l'efficacité de notre algorithme !
Suivant la même philosophie, nous allons encore améliorer notre algorithme. Que se passe-t-il si nous découvrons une lettre du motif dans le texte mais que celle-ci **est mal placée** ?

❐ — À faire vous-même **4** —

On s'intéresse au texte : $\mathcal{T} =$ `"abracadabracadabricadabra"`
et au motif $\mathcal{M} =$ `"adabrica"`.

- ❖ Quand le décalage $s = 5$, a-t-on une occurrence du motif ?

De cet échec, on apprend que le caractère numéro 10 dans $\mathcal{T}$ est un "a". Or, nous savons où sont situés les "a" dans le motif. Nous pouvons donc essayer d'aligner les "a" du motif sur le "a" numéro 10 du texte.



- ❖ Étudions le motif : quels sont les trois valeurs de j où sont situées les "a" du **motif** ?
- ❖ En utilisant l'équation entre s, j et la position du "a" dans le texte, en déduire les trois valeurs du décalage s permettant d'aligner le "a" numéro 10 avec le "a" du motif.
- ❖ Que pouvez-vous dire des valeurs extrêmes de j ?
- ❖ Gagnerait-on du temps si on commençait notre comparaison de chaines par le fin du motif plutôt que par le début ?

Récapitulatif :

Qu'avons-nous appris de ces deux cas ? Nous pouvons récapituler cela en deux points :

Cas 1 : si au décalage s, toutes les lettres du texte correspondent au motif.

- ❖ on a trouvé une occurence du motif, on l'affiche et on se décale d'une unité ($s \leftarrow s + 1$).

Cas 2 : si au décalage s, la j-ième lettre du motif ne correspond pas à la $(s + j)$-ième lettre du texte (on pose $s + j = k$) :

- ❖ la lettre n'appartient pas du tout au motif. On saute tout le texte jusqu'à la (k+1)-ième lettre ($s \leftarrow k + 1$) ;
- ❖ la lettre appartient au motif. On se décale alors du nombre de lettres nous permettant d'aligner la lettre avec la même lettre du motif. On ajoute à s la longueur m du motif et on retranche le décalage pour trouver la lettre identique ($s \leftarrow s + m - \text{decalage} - 1$) .

❐ — À faire vous-même **5**—

On dispose du programme protoBoyerMoore.py sur https://bouillotvincent.github.io et on souhaite le compléter à l'aide du récapitulatif de la page précédente.

- ❖ À la main, donnez toutes les étapes permettant de trouver le motif "`bra`" dans "`bricabrac`".
- ❖ Complétez les fonctions `decalage2` et `checkLetter2` afin de rendre le programme fonctionnel.
- ❖ Grâce à votre programme, affichez toutes les étapes permettant de trouver le motif "`bra`" dans "`bricabrac`". On remarquera en particulier l'opération utilisée sur la lettre B.
- ❖ Appliquez votre programme sur l'exemple sur l'ADN, toujours en affichant les décalages, la valeur de j et les lettres qui ont été décalées. Que pouvez-vous dire de ces décalages ?

Cette nouvelle stratégie permet d'accélérer le traitement car elle permet de sauter des étapes de calcul. On a donc déjà trouvé un algorithme sous-linéaire. Toutefois, dans notre algorithme, les décalages sont calculés à chaque fois alors qu'ils ne dépendent que du motif et de la valeur de j dans le motif. On voit par exemple que certaines valeurs n'ont pas été explorées.

Exercice :
Comment faire pour éviter le calcul systématique de ces décalages?

---

2) Précalcul : mise en place de la table de positions

Dans l'algorithme de Boyer-Moore, une étape de pré-traitement est donc utilisée. À partir du motif $\mathscr{M}$ et uniquement à partir de ce motif, on va créer une table de positions à double entrée : d'une part, l'indice j du caractère du motif qui diffère et d'autre part le caractère c du texte. La valeur du tableau correspondante à deux entrées est l'indice de l'occurrence du caractère c le plus à droite dans le motif avant l'indice j.

Exemple : Avec le motif CGGCAG sur les chaines d'ADN.

Imaginons un texte :

$\mathscr{T}$		?	?	?	C	C	A	G	A	C	?	?	
$\mathscr{M}$		C	G	G	C	A	G						
j =		0	1	2	3	4	5						

Pour les indices j allant de 5 à 3, il y a correspondance entre le motif et le texte. Quand les lettres se correspondent, on ne fait pas de décalage.

Que se passe-t-il maintenant pour j=2 ? Le motif ne correspond plus au texte mais nous savons toutefois que la lettre C du texte existe dans la partie gauche du motif. Le C est en **position** 0. On va pouvoir décaler notre motif de 2-0 = 2 rangs vers la droite et recommencer les comparaisons.

$\mathscr{T}$		?	?	?	C	C	A	G	A	C	?	?	
$\mathscr{M}$				C	G	G	C	A	G				
j =				0	1	2	3	4	5				

Nous venons de calculer le décalage pour j=2 et la lettre C en nous appuyant sur la **position** de la lettre C. Recommençons sur le tableau ci-dessus avec j=5. Le motif ne correspond pas au texte mais nous savons que la lettre C existe dans la partie gauche du motif. Le C le plus proche est en position 3. On va devoir décaler notre motif de 5-3 = 2 rangs vers la droite et recommencer les comparaisons.

Exercice :
Que peut-on dire de la **position** de la lettre C la plus proche en fonction de j ? de G ? de A ?

$\mathscr{T}$		?	?	C	C	A	G	A	C	?	?	?	
				$\mathscr{M}$	C	G	G	C	A	G			
				j =	0	1	2	3	4	5			

On résumera les positions dans un tableau qui prendra la forme suivante :

	C	G	A
0			
1			
2			
3			
4			
5			

Rem : Que faire quand j=3 et que la lettre du texte est un C ? C'est un cas particulier qui n'arrive jamais en raison du mode de fonctionnement de l'algorithme. Toutefois, par définition, on indique alors la position de la lettre C la plus proche sur la gauche. Ici, la position sera donc 0.

À faire vous-même 6

- Construisez à la main la table de Boyer-Moore pour le motif "banane". Comparez votre résultat avec celui de votre voisin et expliquez les éventuelles différences.
- Construisez à la main la table de Boyer-Moore pour un motif d'au moins 5 lettres et connu de vous seul. Puis, donnez cette table à votre voisin qui doit à présent retrouver votre motif initial à partir de la table de Boyer-Moore.
- Si vous avez des doutes, demandez à votre professeur.

❐ — À faire vous-même **7** —

On vous propose le programme Python ci-dessous permettant de calculer la table des positions :

```
1 def tableBM(m):
2     table = [ {} for _ in range(len(m))]
3     for j in range(len(m)):
4         for k in range(j):
5             table[j][m[k]] = k
6     return table
```

- Quelle structure de données est utilisée pour représenter la table ? Expliquez précisément.
- Expliquez ce que font les lignes 3 et 4. En particulier, pourquoi précise-t-on range(j) pour la variable k ?
- Expliquez précisément le fonctionnement de la ligne 5. Pour répondre à cette question, on commencera par s'intéresser à la valeur de m[k] en fonction de k.

À partir de cette table de position, il est facile de calculer le décalage voulu : à partir de la position k et de la valeur de j, le **décalage** s'obtient en calculant j-k.

3) Algorithme de Boyer-Moore

❐ — À faire vous-même **8** —

On vous propose le programme Python page suivante, qui est une implémentation possible de l'algorithme de Boyer-Moore.

```
1  def decalage(table, j, lettre):
2      """ utilise la table table lorsque le caractère
3      numéro j est lettre au lieu du caractère attendu"""
4      if lettre in table[j]:
5          return j - table[j][lettre]
6      else:
7          return j+1
8
9  def rechercheBM(m, t):
10     """affiche toutes les occurrences de m dans t
11     avec l'algorithme de Boyer-Moore"""
12     table = tableBM(m)
13     s = 0
14     while s <= len(t) - len(m):
15         dec = 0
16         for j in range(len(m)-1, -1, -1):
17             if t[s+j] != m[j]:
18                 dec = decalage(table, j, m[j])
19                 break
20         if dec == 0:
21             print("occurrence à la position", s)
22             dec = 1
23         s += dec
```

- Comparez ce programme au programme réalisé au "À faire vous-même 5". Quelles sont les différences et les points communs ?
- Expliquez ce que fait la ligne 12.
- Rappelez l'intérêt d'utiliser une boucle while plutôt qu'une boucle for à la ligne 14
- Expliquez les lignes 16 à 19. En particulier, on se demandera comment fonctionne la fonction decalage(table, j, lettre).
- Que nous indique les lignes 20 à 22 ? En particulier, quel est le rôle joué par dec ?

Conclusion : à partir du travail réalisé ci-dessus, donnez les différentes étapes de l'algorithme de Boyer-Moore.

Exercice :
Appliquez l'algorithme de Boyer-Moore au cas suivant :

texte :
CAATGTCTGCACCAAGACGCCGGCAGGTGCAGACCTTCGTTATAGGCGATGATTTCGAA
CCTACTAGTGGGTCTCTTAGGCCGAGCGGTTCCGAGAGATAGTGAAAGATGGCTGGGCT
GTGAAGGGAAGGAGTCGTGAAAGCGCGAACACGAGTGTGCGCAAGCGCAGCGCCTTA
GTATGCTCCAGTGTAGAAGCTCCGGCGTCCCGTCTAACCGTACGCTGTCCCCGGTACAT
GGAGCTAATAGGCTTTACTGCCCAATATGACCCCGCGCCGCGACAAAACAATAACAGTTT

motif : ACCTTCG

Algorithme de Boyer-Moore :

L'algorithme de Boyer-Moore utilise un retraitement du motif m à chercher dans un texte t pour accélérer cette recherche. Son principe est le suivant :

On teste l'occurrence du motif dans le texte à des décalages s de plus en plus grandes, en partant de s=0.
Pour une position s donnée, on va comparer les caractères de m et de t de la droite vers la gauche : on compare donc $m[M-1]$ avec $t[s+M-1]$, $m[M-2]$ avec $t[s+M-2]$…

Deux possibilités :

1. Si tous les caractères coïncident, on a trouvé une occurrence. On ajoute +1 au décalage s.
2. Sinon, appelons j l'indice de la première différence, c'est à dire le plus grand entier tel que $0 \leq j < M$ et $m[j] \neq t[s+j]$. On appelle c le caractère $t[s+j]$. Nous allons pouvoir sauter des chaines de caractère afin d'accélérer la recherche.
   - si le caractère c **est présent** dans le motif, on ajoute j-k au décalage s, où k est le plus grand entier tel que $0 \leq k < j$ et $m[k] = c$.
   - si le caractère c **n'est pas présent** dans le motif, on ajoute j+1 au décalage s.