1 Introduction

Le principe de la recherche textuelle consiste à trouver la position d'un motif dans un texte. C'est le Ctrl + F de la plupart des éditeurs de texte par exemple.

Par exemple, à quelle position trouve-t-on le motif *NSI* dans le texte *La NSI, c'est mortel !* La réponse est 3.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
L	a		N	S	I	,		c	'	e	s	t		m	o	r	t	e	l	!

2 Algorithme naïf

L'algorithme naïf consiste à parcourir tout le texte du début à la fin, en cherchant pour chaque position possible du motif s'il y a coïncidence avec les caractères du texte et ceux du motif. On commence par « placer » le motif au début du texte et on compare les caractères 2 à 2 entre le motif et le texte. Si 2 caractères entre le motif et le texte ne coïncident pas, on décale le motif d'un cran, et on recommence les comparaisons. Ceci se répète jusqu'à trouver une coïncidence complète ou jusqu'à ce qu'on atteigne la fin du texte.

Exemple du monde biologique : recherche de motif dans l'ADN.
On cherche l'enchaînement CGCAT dans CGGATTCGTATCGCATTTCG.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
C	G	G	A	T	T	C	G	T	A	T	C	G	C	A	T	T	T	C	G
C	G	C	A	T															
	C	G	C	A	T														
		C	G	C	A	T													
			C	G	C	A	T												
				C	G	C	A	T											
					C	G	C	A	T										
						C	G	C	A	T									
							C	G	C	A	T								
								C	G	C	A	T							
									C	G	C	A	T						
										C	G	C	A	T					
											C	G	C	A	T				

`Réponse : 11`

Cet algorithme peut gagner en efficacité avec un peu d'astuce... On va essayer de limiter le nombre de comparaisons.

3 Algorithme de Boyer-Moore (simplifié Horspool)

L'algorithme de Boyer-Moore s'appuie sur les caractéristiques suivantes :
- L'algorithme effectue un **prétraitement** du motif. Cela signifie que l'algorithme « connaît » les caractères qui se trouvent dans le motif.
- On commence la comparaison **par la droite** du motif. Par exemple pour le motif NSI, on compare d'abord le I, puis le S, et enfin le N.
- Dans la méthode naïve, les décalages du motif vers la droite se faisaient toujours d'un seul cran à la fois. L'intérêt de l'algorithme de Boyer-Moore, c'est qu'il permet, dans certaines situations, d'effectuer un **décalage de plusieurs crans en une seule fois**.

Exemple : recherche de CGCAT dans CGGATTCGTATCGCATTTCG :

L'algorithme de Boyer-Moore effectue la vérification à l'envers : pour chercher CGCAT dans le texte, on teste d'abord le 5ème caractère du texte. Si c'est un T, on compare si le 4ème caractère est un A, puis le 3ème un C, ... jusqu'au 1er caractère.
L'intérêt de cet algorithme est que si on trouve en 5ème position un caractère n'apparaissant pas dans le motif CGCAT, alors on sait que ce motif commence au mieux en 6ème position et on saute directement à la 10ème position pour rechercher un T.
Que faire si on trouve par exemple un G au lieu d'un T en 5ème position ?
Il se peut que ce soit le G de CGCAT, auquel cas le T serait en 8ème position. L'algorithme saute alors à la 8ème position pour rechercher le T (puis éventuellement le A, le C...).
Le saut dépend donc du caractère lu : dans le cas de CGCAT, +5 si le caractère n'est pas dans le motif, +3 pour un G, +2 pour un C, +1 pour un A.

Plutôt que de calculer la longueur du saut à chaque fois, on commence donc par un **prétraitement** du motif pour construire une **table de sauts** avant de lancer la recherche.

Tableau de sauts :

caractère erroné dans le texte	base de saut
A	1
C	2
G	3
Autre	5

Attention ! : ce saut « de base » doit être réduit du nombre de caractères qui coïncident.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
C	G	G	A	T	T	C	G	T	A	T	C	G	C	A	T	T	T	saut effectif
C	G	C	A	T														(G) 3 - 2 = 1
	C	G	C	A	T													(T) 5 - 1 = 4
					C	G	C	A	T									(A) 1 - 0 = 1
						C	G	C	A	T								(T) 5 - 2 = 3
									C	G	C	A	T					(C) 2 - 0 = 2
											C	G	C	A	T			

Attention : Cas particulier :
Chercher "tata" dans "tatie est battante" :

Tableau de sauts :

caractère erroné dans le texte	base de saut
t	1
Autre	4

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
t	a	t	i	e		e	s	t		b	a	t	t	a	n	t	e	saut effectif
t	a	t	a															(i) 4 - 0 = 4
				t	a	t	a											(s) 4 - 0 = 4
								t	a	t	a							(b) 4 - 1 = 3
											t	a	t	a				(t) 1 - 2 = **-1**
										t	a	t	a					(t) 1 - 0 = 1
											t	a	t	a				(t) 1 - 2 = **-1**

Il se peut que le calcul du saut effectif donne un résultat nul ou négatif (le motif « n'avance » plus... !), alors dans ce cas il faut forcer un saut de 1 cran, sinon l'algorithme boucle à l'infini...

4 Proposition d'implantation en Python

```
def pretraitement(motif):
    table = dict()
    for i in range(65537): # ça couvre beaucoup de caractères...
        table[chr(i)] = len(motif)
    for i in range(len(motif)-1):
        table[motif[i]] = len(motif) - i - 1
    return table

def recherche(motif, texte):
    table = pretraitement(motif)
    i = 0 # indice parcourant le texte
    while i <= len(texte) - len(motif):
        j = len(motif) - 1 # indice parcourant les coïncidences texte/motif
        while j >= 0 and texte[i + j] == motif[j]:
            j = j - 1
        if j == -1:
            return i
        else:
            i = i + max(1, table[texte[i + j]] - (len(motif) - 1 - j))
    return -1
```