

Recherche textuelle

Fiche de synthèse

① LE PROBLÈME

On cherche toutes les positions d'un **motif** de longueur m dans un **texte** de longueur n , avec $m \leq n$. Le motif 'bra' apparaît dans 'abracadabra' aux positions 1 et 8. Ce problème intervient dans les éditeurs de texte, les moteurs de recherche, la commande `grep` et l'analyse de l'ADN.

② ALGORITHME NAÏF

On essaie chaque position i de 0 à $n - m$ et l'on compare le motif au texte, caractère par caractère, de gauche à droite ; au premier caractère différent, on passe à la position suivante.

```
def algo_naif(texte, motif):
    n = len(texte)
    m = len(motif)
    indices = []
    for i in range(n - m + 1):
        trouve = True
        for j in range(m):
            if texte[i + j] != motif[j]:
                trouve = False
                break
        if trouve:
            indices.append(i)
    return indices
```

La borne est $n - m + 1$ car $n - m$ est la dernière position où le motif tient dans le texte. Il y a $n - m + 1$ positions et au plus m comparaisons pour chacune : la complexité dans le pire cas est en $O(n \times m)$, quadratique quand m est de l'ordre de n . Chercher 'aab' dans 'aaaaaaaa' demande presque trois comparaisons à chaque position.

③ ALGORITHME DE BOYER-MOORE-HORSPOOL

- On compare le motif au texte **de droite à gauche**, en commençant par la dernière lettre du motif.
- Après un échec, on décale le motif d'après le caractère du texte aligné avec la **fin** du motif : on l'aligne avec l'occurrence la plus à droite de ce caractère dans le motif, ou l'on décale de m si le motif ne le contient pas.

Recherche de 'dab' dans 'abracadabra' ; la dernière lettre du motif, encadrée, est comparée en premier à la lettre du texte située au-dessus :

0	1	2	3	4	5	6	7	8	9	10
a	b	r	a	c	a	d	a	b	r	a
d a b										
r ≠ b, r absent : +3										
d a b										
a ≠ b : +1										
d a b										
d ≠ b : +2										
d a b										
trouvé en 6										

On teste 4 positions au lieu de 7 pour trouver cette occurrence.

Définition 1 (Table de décalages) Pour chaque caractère c du motif, sauf le dernier, le **décalage** vaut $m - 1 - i$, où i est la position la plus à droite de c dans le motif, dernière position exclue. Un caractère absent du motif donne le décalage m .

Pour 'dab', la table est {'d': 2, 'a': 1}; pour 'maman', {'m': 2, 'a': 1}. On exclut le dernier caractère pour ne jamais obtenir un décalage nul, qui ferait boucler l'algorithme.

```
def pre_traitement(motif):
    m = len(motif)
    decalages = {}
    for i in range(m - 1):
        decalages[motif[i]] = m - 1 - i
    return decalages

def boyer_moore(texte, motif):
    n = len(texte)
    m = len(motif)
    decalages = pre_traitement(motif)
    indices = []
    i = 0
    while i <= n - m:
        j = m - 1
        while (j >= 0
              and texte[i + j] == motif[j]):
            j -= 1
        if j < 0:
            indices.append(i)
            i += 1
        else:
            c = texte[i + m - 1]
            if c in decalages:
                i += decalages[c]
            else:
                i += m
    return indices
```

Parcourir le motif de gauche à droite garde, pour chaque caractère, sa position la plus à droite : chaque nouvelle occurrence remplace la précédente.

④ COMPLEXITÉ

- **Meilleur cas** : les caractères du texte sont absents du motif, on décale de m à chaque fois. Il y a environ n/m comparaisons : la complexité est **sous-linéaire**, on trouve le motif sans lire tout le texte.
- **Pire cas** : $O(n \times m)$, comme l'algorithme naïf ; chercher 'baa' dans 'aaaaaa' décale toujours d'une position.
- **En pratique**, sur un texte en français, les décalages sont grands : l'algorithme est souvent trois à cinq fois plus rapide que l'algorithme naïf, d'autant plus que le motif est long.

À retenir.

- L'algorithme naïf teste chaque position, de gauche à droite : $O(n \times m)$ dans le pire cas.
- Boyer-Moore-Horspool compare de droite à gauche et saute des positions grâce à une table de décalages calculée à l'avance.
- Le décalage dépend du caractère du texte aligné avec la fin du motif.
- Dans le meilleur cas, la recherche demande environ n/m comparaisons.