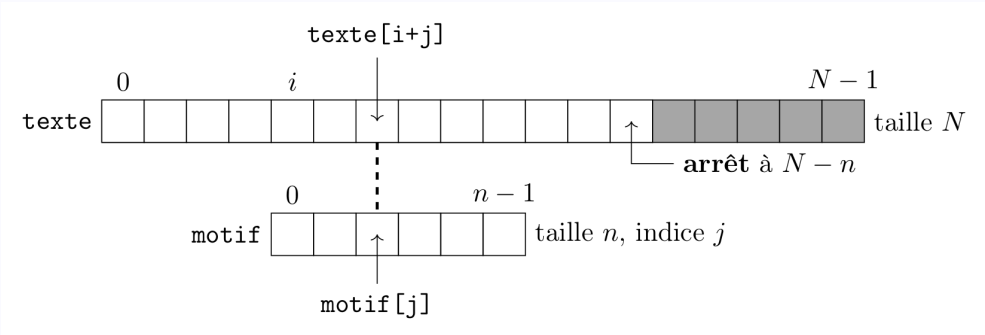
TD 7 : Recherche d'une séquence textuelle, recherche des deux valeurs les plus proches

Exercice 1 (Recherche d'une séquence textuelle)

Fonction auxiliaire : En python, il est possible de créer à l'intérieur d'une fonction, une fonction auxiliaire comme par exemple dans le cas du tri par sélection.

```
def tri_selection(T):
    def minimum():  ##################
        pos,mini=j,T[j]              #
        for k in range(j+1,len(T)):  # fonction
            if T[k]<mini :           # auxiliaire
                pos,mini = k,T[k]    #
        return pos  ##################
    for j in range(len(T)-1):
        i=minimum()
        if i!= j:
            T[i], T[j] = T[j], T[i]
    return T
```

On cherche ici de manière naïve si une chaîne de caractère `texte` contient une sous chaîne de caractère donnée `motif`. Pour cela, on regardera le schéma suivant :



1. Écrire une fonction `recherche_naive(motif,texte)` qui prend en entrée deux chaînes de caractère et renvoie `True` si `motif` est une sous chaîne de caractère de `texte`, et `False` sinon. On créera pour cela une fonction auxiliaire `occurence()` qui teste si la sous chaîne de `texte` comprise entre les indices i et $i+n-1$ est identique à `motif` (n correspondant à la taille de `motif`).
2. Tester cette fonction sur le texte de votre choix (d'une grande taille).

Exercice 2 (Recherche des deux valeurs les plus proches)

1. Écrire une fonction `Min_et_max(L)` qui renvoie le tuple constitué du min et du max d'une liste de flottant L en ne parcourant qu'une fois la liste.

   Le mot clef `assert` doit être suivi d'un booléen, et (facultatif) d'une virgule et d'une chaîne de caractère : si le booléen est Faux alors la fonction s'arrête et le message s'affiche comme dans :

```
def test(n):
    assert n%2==0, "donner un nombre pair"
    return n//2
```

```
>>> test(11)
Traceback (most recent call last):
  File "<console>", line 1, in <module>
  File "<tmp 1>", line 15, in test
    assert n%2==0, "donner un nombre pair"
AssertionError: donner un nombre pair
```

2. Écrire une fonction `cherche_proche(T)` qui renvoie les deux valeurs les plus proches d'une liste d'entiers. On vérifiera que la liste contienne au moins deux éléments à l'aide du mot-clé `assert`.
   Indice : On utilisera une variable `dist` *qui sera initialisée par la plus grande distance entre deux flottants de la liste.*

Exercice 3 (Défi : 8 dames sur un échiquier)

On cherche à mettre huits reines sur un échéquiers sans que celles-ci ne puissent se prendre en un coup. Écrire une fonction qui affiche les positions des huits dames, de préférence sous la forme d'un échéquier. On pourra utiliser le module itertools et la fonction permutations :
`from itertools import permutations`
`permutations(liste)` renvoie toutes les permutations de `liste`
`permutations(range(3))` renvoie [(0,1,2),(0,2,1),(1,0,2),(1,2,0),(2,0,1),(2,1,0)] (par forcément dans cet ordre).

On rappelle qu'un échiquier est un plateau de jeu ayant 8 lignes (numérotées de 1 à 8, 1 du côté des blancs) et 8 colonnes (numérotées par une lettre de `a` à gauche pour les blancs à `h`). Pour les blancs, la case à l'extrémité droite de la première ligne est blanche. Une case est donc repérée par la donnée d'une lettre et d'un numéro comme `e4`.
On rappelle aussi qu'une dame *mange* toute pièce située sur la même ligne, la même colonne ou les mêmes diagonales.