
	<b>Séquence 9</b>	<b>1 / 2</b>
	<b>Recherche séquentielle</b> <b>Tableau trié</b>	

## 1 Objectif

Optimiser la recherche d'un élément dans un tableau (ou liste).

## 2 Problème posé

Nous souhaitons rechercher l'élément 5 dans le tableau trié  $T = [1, 2, 5, 9, 10, 14, 17, 24, 41]$ .

Avec une recherche séquentielle ou recherche par balayage, on parcourt la liste du début à la fin en comparant chaque valeur à l'élément recherché.

Dans le pire de cas, on parcourt la liste entièrement (cas où l'élément recherché est en dernière position).

## 3 Principe de la recherche séquentielle dans un tableau trié

### 3.1 Principe général

Etant donné un tableau d'entiers, il s'agit de rechercher une valeur donnée dans ce tableau et, si elle s'y trouve, de retourner le plus petit indice correspondant à cette valeur. Si la valeur ne s'y trouve pas, on peut retourner None, par exemple.

**Exemple** : `tab_trie = [ 3, 6, 15, 7 , 23, 7, 1]`

La recherche de la valeur 15 doit retourner 2.

La recherche de la valeur 7 doit retourner 3.

La recherche de la valeur 5 doit retourner None.

### 3.2 Principe optimisé

Si le tableau est trié dans l'ordre croissant, on peut s'éviter de parcourir le tableau si la valeur recherchée est :

- plus petite que le premier élément du tableau.
- plus grande que le dernier élément du tableau.
- plus petite que la valeur du tableau à l'index considéré (ou traité)

**Cas a)** : `tab_trie = [1, 3, 4, 7, 7, 9, 18, 23]`.

- Valeur cherchée : 0
- Valeur minimale : 1
- Valeur maximale : 23

La valeur 0 n'est évidemment pas dans ce tableau car  $0 < 1$ , premier élément de `tab_trie`.

**Cas b)** : `tab_trie = [1, 3, 4, 7, 7, 9, 18, 23]`.

- Valeur cherchée : 24
- Valeur minimale : 1
- Valeur maximale : 23

La valeur 24 n'est évidemment pas dans ce tableau car  $24 > 23$ , dernier élément de `tab_trie`.

**Cas c)** : `tab_trie = [1, 3, 4, 7, 7, 9, 18, 23]`.

- Valeur cherchée : 8
- Valeur minimale : 1
- Valeur maximale : 23

La valeur 8 n'est évidemment pas dans ce tableau car  $8 < 9$ , élément à l'index 5 de `tab_trie`.

## 4 Implémentations

### 4.1 Version boucle while

**Question 1.** A l'aide du descriptif donné précédemment, **compléter** le programme suivant puis **implémenter** l'algorithme.

```
def rech_tri_w(val, T): #retourne le plus petit rang de val dans le
                        #tableau T trié ou None
    i=0
    n=len(T)
    if .....:
        .....

    elif .....:
        .....
    else:
        while .....:
            .....
```

**Question 2.** **Tester** la fonction sur le tableau tab\_trie de l'exemple, avec différentes valeurs. **Afficher** les valeurs pertinentes (aide au débogage, valeur cherchée, finale...)

### 4.2 Version boucle for

**Question 3.** **Modifier** puis **tester** le programme précédent, après avoir remplacé la boucle while par une boucle for.