

5.2

Recherche de valeurs

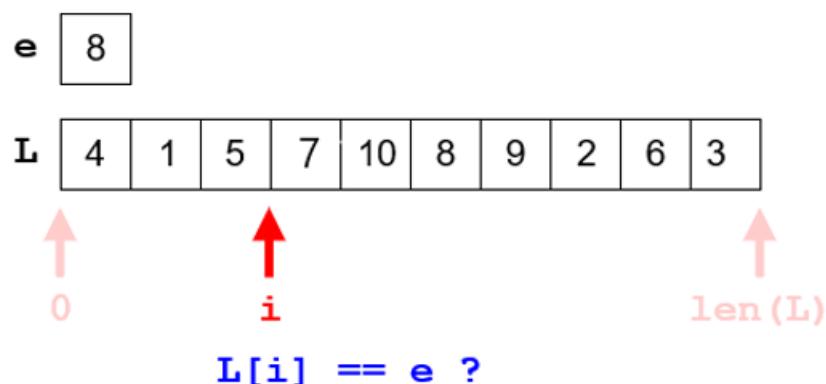
NSI 1ÈRE - JB DUTHOIT

5.2.1 Algorithme de recherche de valeurs dans un tableau

La recherche d'une valeur consiste ici à déterminer, s'il existe, l'indice i d'un élément e dans un tableau L .

- ☛ Si e n'est pas présent dans L , on renvoie FAUX.
- ☛ Si e est présent plusieurs fois dans L , on renvoie seulement l'indice le plus petit.

Exemple : Je recherche si la valeur 8 est dans la liste L :



Exercice 5.1

Écrire un algorithme de recherche d'une valeur utilisant une boucle POUR.

L'algorithme est une fonction qui prend en argument un tableau d'entiers et une valeur entière, et qui renvoie l'indice le plus petit de cette valeur (FAUX si la valeur n'est pas présente dans le tableau).

Exercice 5.2

Écrire un algorithme de recherche d'une valeur en utilisant une boucle TANT QUE.

L'algorithme est une fonction qui prend en paramètres un tableau d'entiers et une valeur entière, et qui renvoie le plus petit indice de cette valeur (FAUX si la valeur n'est pas présente dans le tableau)

Exercice 5.3

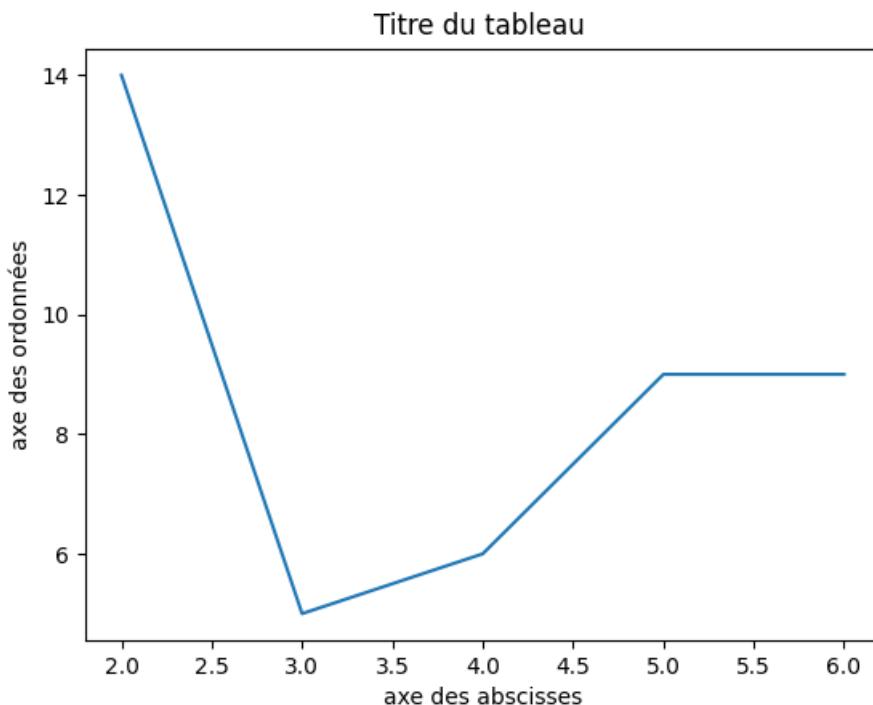
1. Créer l'algorithme de la fonction `NbOccurrences(T,L)` qui reçoit en argument un tableau d'entiers T et une liste d'entiers L , et qui retourne le nombre d'occurrences de e dans T .
 - ☛ A noter que la fonction renvoie -1 si e n'est pas dans T .
 - ☛ Faire une version avec POUR et une autre avec TANT QUE...
2. Implémenter cet algorithme en Python

5.2.2 Durée d'exécution

Exercice 5.4

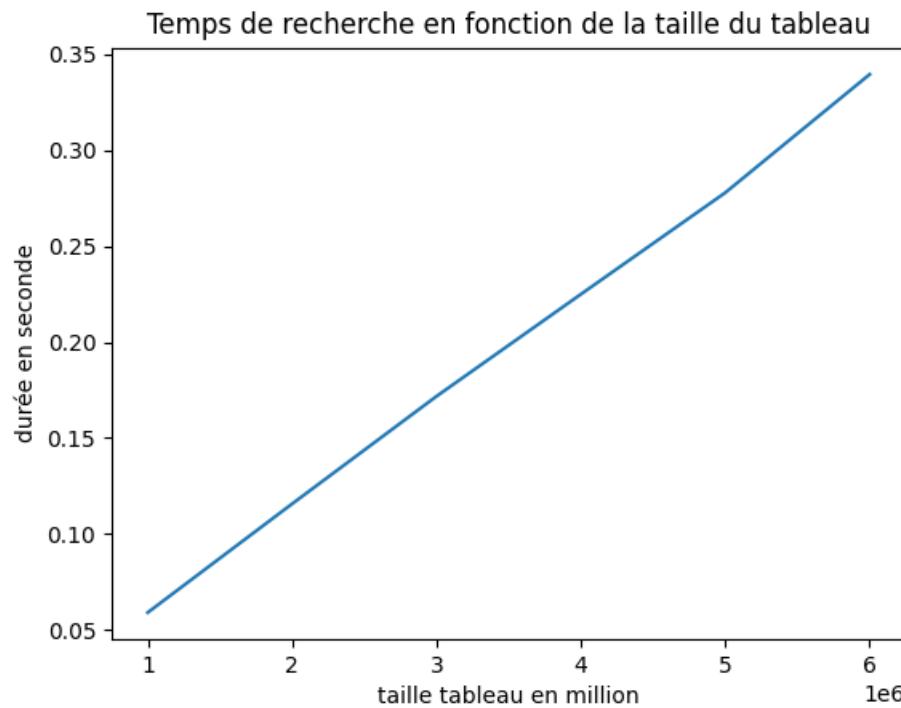
1. Construire un tableau de 1 million d'éléments, avec des entiers de 0 à 999999.
2. Appliquer un des algorithmes précédents à cette liste et calculer le temps d'exécution pour la recherche d'un élément qui n'est pas dans le tableau (afin d'être sûr qu'on parcourt le tableau en entier). Garder la valeur en mémoire.
3. Recommencer les deux questions précédentes pour une liste de 2 000 000 , puis 3 000 000 etc jusqu'à 6 000 000. Garder les résultats.
4. Observer ce code qui permet l'affichage du graphique ci-après.

```
import matplotlib.pyplot as plt
plt.title('Titre du tableau ')
plt.plot([2,3,4,5,6],[14,5,6,9,9])
plt.xlabel('axe des abscisses')
plt.ylabel('axe des ordonnées')
plt.show()
```



Exemple d'utilisation de matplotlib

5. Placer les points obtenus aux questions précédentes sur un graphique, afin de visualiser la durée d'exécution de la fonction recherche en fonction de la taille de la liste.
Vous devriez obtenir un graphique comme celui-ci :



On en déduit, de façon empirique, que le temps de recherche est linéaire par rapport à la taille du tableau.

5.2.3 Complexité temporelle

Nous allons essayer de formaliser la notion précédente et d'estimer le coût en temps de cet algorithme.

☞ Nous supposerons que chaque opération (affectation, comparaison, opération arithmétique, ...) a un coût de 1 « unité » de temps.

Reprendons un des deux algorithmes, celui avec la boucle pour par exemple :

```

1 FONCTION OCCURRENCE(T :tableau d'entiers, element :entier)
2   POUR i DE 0 A longueur(T)-1 FAIRE
3     SI T[i] = element ALORS
4       RENVOYER i
5   RENVOYER False

```

On suppose que la taille du tableau est n :

ligne 2 : au pire n affectations pour i plus 1 interrogation de longueur de tableau

ligne 3 : au pire, n comparaisons

Soit $T(n)$ le nombre d'opérations. $T(n) = 2n + 1$. On a donc une complexité en $\mathcal{O}(n)$, soit une complexité linéaire.

5.2.4 Parcours de tableau python

Exercice 5.5

1. Créer l'algorithme de la fonction **maximum(tab)** qui reçoit en argument un tableau d'entiers et qui retourne le maximum du tableau.
2. Calculer la complexité temporelle de cette algorithme.
3. Implémenter cet algorithme en Python

Exercice 5.6

1. Créer l'algorithme de la fonction **minimum(T)** qui reçoit en argument un tableau d'entiers et qui retourne le minimum du tableau.
2. Calculer la complexité temporelle de cet algorithme.
3. Implémenter cet algorithme en Python

Exercice 5.7

1. Créer l'algorithme de la fonction **moyenne(tab)** qui reçoit en argument un tableau d'entiers et qui retourne la moyenne des éléments du tableau.
2. Calculer la complexité temporelle de cet algorithme.
3. Implémenter cet algorithme en Python

Exercice 5.8

Étude des températures enregistrées à Lesquin en Aout 2020.

Vous disposez d'un tableau nommée température comportant les températures relevées sous abri chaque jour d'Aout 2020 au plus chaud de la journée pour la ville de Lesquin relevées sur le site : www.infoclimat.fr/observations-meteo/temp-reel/lille-lesquin/07015.html :

```
Temperature = [27.4 , 24.1 , 22.2 , 24.6 , 29.8 , 33.2 ,
35.1 , 36.9 , 36.0 , 34.2 , 34.5 , 33.7 , 30.6 ,
23.1 , 26.9 , 26.4 , 23.8 , 25.1 , 26.1 , 30.9 ,
28.1 , 21.8 , 20.6, 23.1, 23.3, 20.7, 22.2,
20.9, 17.1, 18.2, 19.4]
```

Votre travail consiste à écrire puis implémenter un algorithme sous la forme de fonctions permettant de déterminer :

1. la température la plus élevée,
2. la température la plus basse,
3. la température moyenne.

Tester votre programme. Recherche dans un dictionnaire

Exercice 5.9

Etude démographique de la communauté de communes Pévèleloises.

Voici un tableau de dictionnaires comportant les noms des communes de la Pévèle, leur superficie et leur nombre d'habitants en Janvier 2020 :

```
communautePevele = [
{'Commune': 'Attiches', 'Population': '2229', 'Superficie': '668'},
{'Commune': 'Avelin', 'Population': '2304', 'Superficie': '1376'},
{'Commune': 'Bachy', 'Population': '1396', 'Superficie': '641'},
{'Commune': 'Bersée', 'Population': '2120', 'Superficie': '1093'},
{'Commune': 'Bourghelles', 'Population': '1418', 'Superficie': '655'},
```

```
{'Commune': 'Camphin-en-Pévèle', 'Population': '1570', 'Superficie': '645'},  

{'Commune': 'Cappelle-en-Pévèle', 'Population': '1959', 'Superficie': '811'},  

{'Commune': 'Cobrieux', 'Population': '518', 'Superficie': '284'},  

{'Commune': 'Cysoing', 'Population': '4427', 'Superficie': '1362'},  

{'Commune': 'Ennevelin', 'Population': '1973', 'Superficie': '992'},  

{'Commune': 'Genech', 'Population': '2050', 'Superficie': '746'},  

{'Commune': 'Louvil', 'Population': '835', 'Superficie': '250'},  

{'Commune': 'Mérignies', 'Population': '2105', 'Superficie': '861'},  

{'Commune': 'Moncheaux', 'Population': '1315', 'Superficie': '497'},  

{'Commune': 'Mons-en-Pévèle', 'Population': '2054', 'Superficie': '1237'},  

{'Commune': 'Mouchin', 'Population': '1340', 'Superficie': '919'},  

{'Commune': 'Templeuve', 'Population': '5778', 'Superficie': '1584'},  

{'Commune': 'Tourmignies', 'Population': '756', 'Superficie': '203'},  

{'Commune': 'Wannehain', 'Population': '839', 'Superficie': '371'}]
```

Votre travail consiste à écrire puis à implémenter un algorithme sous la forme de fonctions permettant de déterminer :

1. la ville la plus peuplée,
2. La ville la moins peuplée.
3. le nombre moyen d'habitants par km² dans la Pévèle.

Tester votre programme

Exercice 5.10

Écrire une fonction `carre(tab)` qui prend en argument `tab`, de type `list`, un tableau d'entiers (de type `int`), et qui renvoie un nouveau tableau avec les carrés des entiers du tableau passé en argument.

Exercice 5.11

Écrire une fonction `double(tableau)` qui prend en argument `tableau`, de type `list` et contenant des entiers (type `int`) et qui renvoie un nouveau tableau où chaque valeur a été doublée :

```
assert double([1,2,3,4,6]) == [2,4,6,8,12]
```

Exercice 5.12

Écrire une fonction `supp(tableau)` qui prend en argument `tableau`, de type `list`, un tableau d'entiers (avec au moins deux éléments, de type `list`), et qui renvoie un nouveau tableau, identique au premier, mais avec les deux premiers termes supprimés.

Exercice 5.13

Écrire une fonction `signe(tab)` qui prend en argument `tab` (type `list`) un tableau d'entiers (type `int`) et qui renvoie un tuple (`n,p`) où `n` est le nombre d'entiers négatifs et `p` le nombre d'entiers positifs.

► L'entier 0 sera compté dans les positifs.

Exercice 5.14

Écrire une fonction `grandsmots(tab)` qui prend en argument un tableau `tab` (de type `list`) contenant des chaînes de caractères (de type `str`), et qui retourne une autre liste contenant les mots de plus de 5 caractères.

Exercice 5.15

Programme une fonction `rotation(tab)` qui décale d'un rang tous les éléments du tableau `tab` (de type `list`).

- ☛ le dernier élément devenant le premier.
- ☛ La fonction renvoie une nouvelle liste.

Par exemple `rotation([1,2,3,4])` renvoie la liste `[4,1,2,3]`.

Exercice 5.16

Programme une fonction `inverser(tab)` de paramètre `tab` (de type `list`), un tableau d'entiers (de type `int`) et qui renvoie un nouveau tableau avec les éléments dans l'ordre inverse.

Par exemple, `assert inverser([1,2,3,4]) == [4,3,2,1]`.

Exercice 5.17

Programme une fonction `supprimer_rang(liste,rang)` qui renvoie une liste formée de tous les éléments, sauf celui au rang donné. Par exemple, `supprimer_rang([8,7,6,5,4],2) == [8,7,5,4]`.
(l'élément 6 qui était au rang 2 est supprimé)