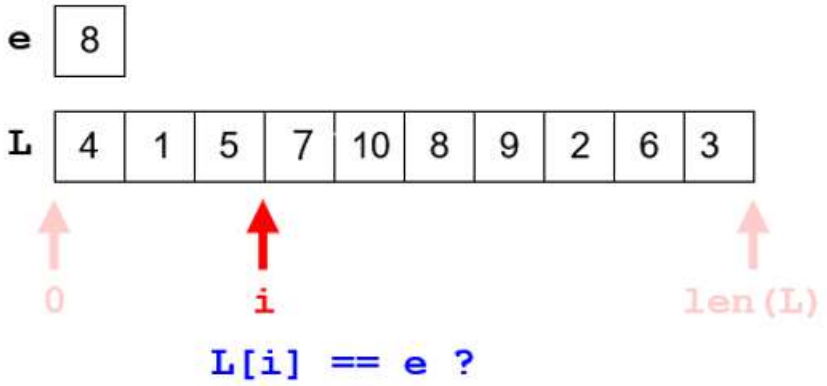
6.2 Recherche d'occurrences

NSI 1ère - JB Duthoit

6.2.1 Algorithme de recherche d'occurrences

La recherche d'une occurrence !consiste à déterminer la position (l'indice i) d'un élément e présent dans la liste L. On parle d'occurrence de e dans L.
Exemple : Je recherche l'occurrence de 8 dans la liste L :



Exercice 6.124
Ecrire un algorithme de recherche d'une occurrence 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'occurrence de cette valeur (FAUX si la valeur n'est pas présente dans le tableau).

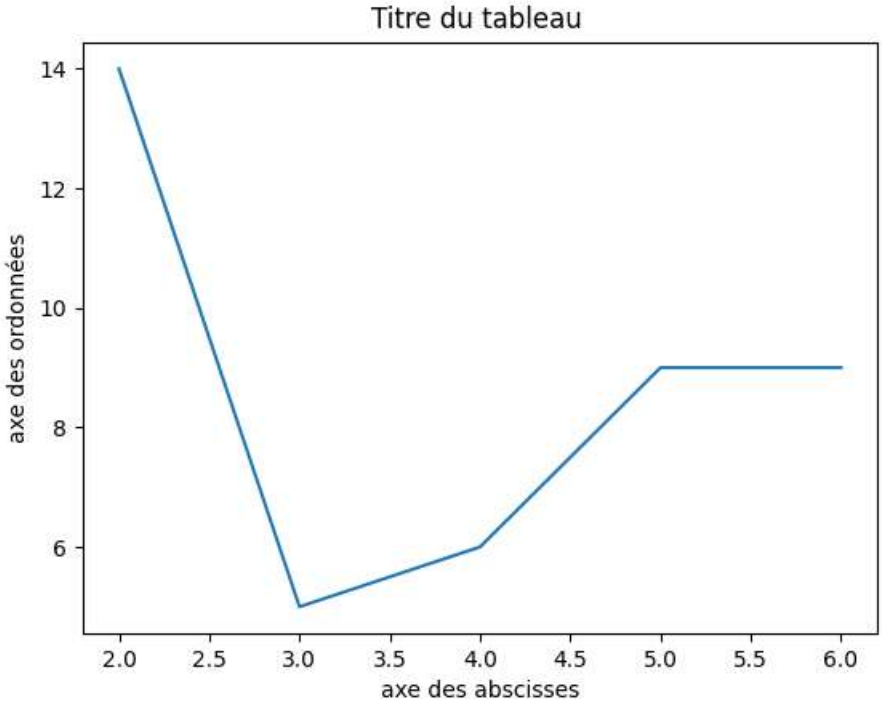
Exercice 6.125
Ecrire un algorithme de recherche d'une occurrence en utilisant une boucle TANT QUE.
L'algorithme est une fonction qui prend en paramètres un tableau d'entiers et une valeur entière, et qui renvoie l'occurrence de cette valeur (FAUX si la valeur n'est pas présente dans le tableau)

6.2.2 Durée d'exécution

Exercice 6.126

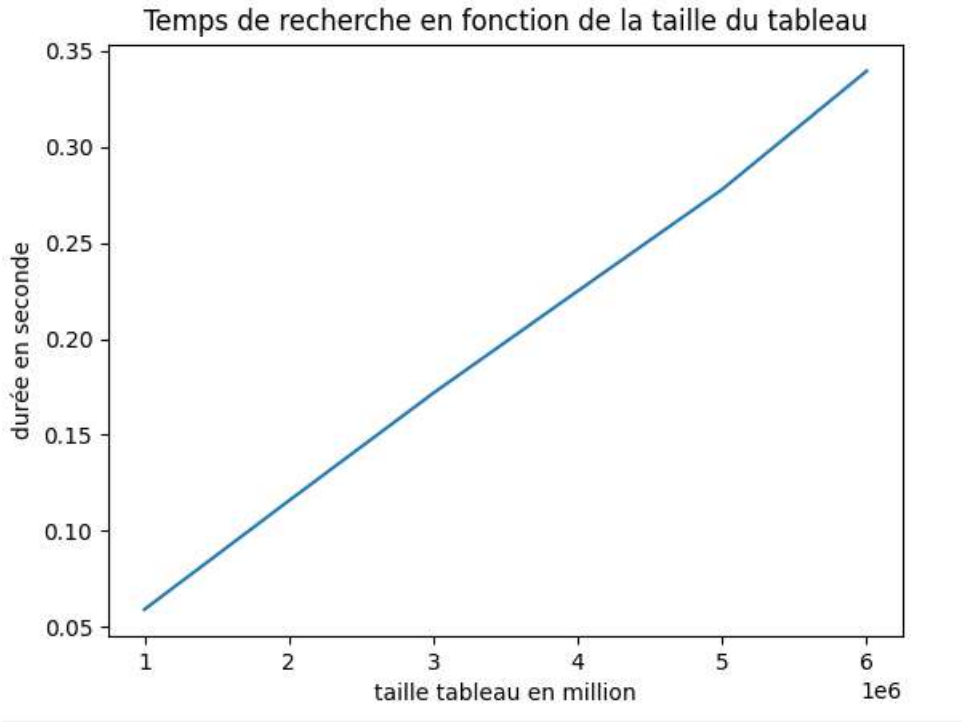
1. Construire une liste de 1 million d'élements, avec des entiers de 0 à 999999.
2. Appliquer un des algorithmes précédents à cette liste et calculer le temps d'exécution pour la recherche d'un élément qui n'est pas dans le tableau (afin d'être sûr qu'on parcourt le tableau en entier). Garder la valeur en mémoire.
3. Recommencer les deux questions précedentes 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.

6.2.3 Compléxité 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.

Reprenons un des deux algoritmes, celui avec la boucle pour par exemple :

```
1 FONCTION OCCURRENCE(T :tableau d'entiers, element :entier)
2 |  POUR i DE O 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.

Exercice 6.127

1. Créer l'algorithme de la fonction **maximum(T)** 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 6.128

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 cette algorithme.
3. Implémenter cet algorithme en Python

Exercice 6.129

1. Créer l'algorithme de la fonction **moyenne(T)** 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