

## Recherche séquentielle

### 1. Spécification de la fonction recherche

recherche prend pour paramètre une liste de mots.

recherche retourne l'index du mot dans la liste de mots, recherché de manière séquentielle.

Dans la fonction recherche: X est un mot tiré aléatoirement avec `choice(liste_mots)`

on parcourt la liste avec une boucle non bornée `while`, tant que X n'est pas trouvé dans la liste

on augmente la valeur de j à chaque nouvelle itération jusqu'à trouver X dans la liste:

```
1. def recherche_mot(liste_mots):
2.     j = 0
3.     n = len(liste_mots)
4.     X = choice(mots) # a completer
5.     while j<n and X!=liste_mots[j]:
6.         j += 1 # à completer
7.     if j==n : return -1
8.     return
```

### 2. Etude du temps moyen en fonction de la longueur de liste

```
1. def traitement_fichier(fichier):
2.     mots = []
3.     with open(fichier, encoding='utf-8') as f:
4.         for mot in f.read().splitlines():
5.             mots.append(mot)
6.     for i in range(1000):
7.         t0 = time.time()
8.         recherche_mot(mots)
9.         t1 = time.time()
10.        T.append(t1-t0)
11.    r = np.mean(T)
12.    return len(mots),r
13.
14. L = []
15. L.append((0,0))
16. L.append(traitement_fichier('ods4.txt'))
17. L.append(traitement_fichier('gutenberg.txt'))
18. L.append(traitement_fichier('liste_francais.txt'))
19. L.append(traitement_fichier('pli07.txt'))
```

fichier	nombre mots	temps moyen de recherche (s)
liste_francais.txt	21074	0.02331569260093057
pli07.txt	78855	0.02231073593665026
gutenberg.txt	336530	0.024773356865863412
ods4.txt	369085	0.02414901211313958

## 3. Tracé du nuage de points

```

1. import matplotlib.pyplot as plt
2. # creation de listes x,y
3. # a l'aide de la librairie np
4. L = np.array(L)
5. x = L[:,0]
6. y = L[:,1]
7. plt.scatter(x,y)
8. plt.xlabel('N mots')

```

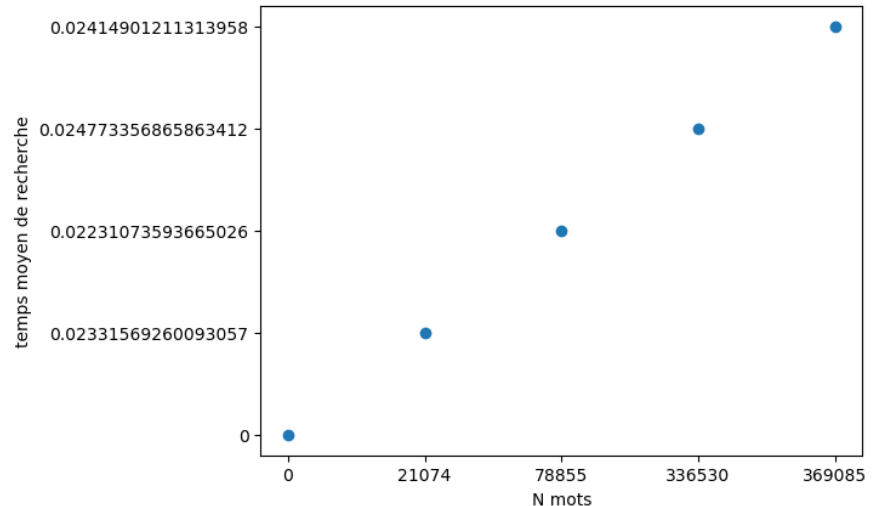
```

>>> L
[(0, 0),
 (21074, 0.02331569260093057),
 (78855, 0.02231073593665026),
 (336530, 0.024773356865863412),
 (369085, 0.02414901211313958)]

```

A. Démontrer que la complexité algorithmique de la recherche séquentielle est  $O(n)$ . Les seules opérations significative seront les opérations de comparaison.

B. Le graphique, confirme-t-il que l'algorithme de recherche est de classe de complexité linéaire?



## Recherche dichotomique

```

1. def recherche_dicho_mot(mots):
2.     gauche = 0
3.     droite = len(mots)
4.     trouve = False
5.     X = choice(mots)
6.     i=0
7.     while gauche <= droite and not trouve:
8.         i+=1
9.         milieu = (gauche+droite)//2
10.        if mots[milieu] == X:
11.            trouve = True
12.        elif mots[milieu] < X:
13.            gauche = milieu + 1
14.        else:
15.            droite = milieu - 1
16.    if not trouve :
17.        return -1
18.    return (milieu,i)

```

A. Donner une spécification de la fonction.

B. Compléter la colonne  $\text{np.log}_2(n)$  dans le tableau.

C. Prouver que la complexité algorithmique est  $O(\log(n))$

L'étude de 2 dictionnaires de mots donne le tableau suivant:

fichier	nombre mots n	$\text{np.log}_2(n)$	nombre itérations	temps moyen de recherche
pli07.txt	78855		15.35	7.83681869506e-06
ods4.txt	369085		17.63	1.20854377746e-05