```python
## exo 1 : recherche
N = [3,6,8,20,8,4,6]
S = 'azertyuiop'

#Q1
def cherche(e,L):
    for i in L:
        if i == e:
            return True
    return False

print(cherche(21,N))
print(cherche('z',S))

## Q2
def cherche(e,L):
    for i in range(len(L)):
        if L[i] == e:
            return i
    return None

print(cherche(21,N))
print(cherche('z',S))

## Q3
def cherche(e,L):
    nbiter = 0
    for i in range(len(L)):
        nbiter += 1
        if L[i] == e:
            return i , nbiter
    return None , nbiter

print(cherche(21,N))
print(cherche('z',S))

## Q4 Complexité dans le pire des cas
"""
Le nb de tests est minimal si e est en première position, et maximal si e est en fin de liste, c
"""

## exo 2 : recherche par dichotomie

# Q1
def cherche_dicho(e,L):
    # cette fonction retourne un booléen indiquant si l'élément e est présent dans la liste trié
    b1 = 0 ; b2 = len(L)
    if e < L[b1] or e > L[b2-1]:
        return False
    else:
        while b1 < b2:
            ind = (b1 + b2) // 2
            if e == L[ind]:
                return True
            elif e < L[ind]:
                b2 = ind
            else:
                b1 = ind + 1
        return False

L1 = [1,3,4,5,6,12,15]
```

```
L2 = list(range(0,20,2))
x = 0
print('valeur',x, ':',cherche_dicho(x,L1))
x = 1
print('valeur',x, ':',cherche_dicho(x,L1))
x = 6
print('valeur',x, ':',cherche_dicho(x,L1))
x = 15
print('valeur',x, ':',cherche_dicho(x,L1))
x = 111
print('valeur',x, ':',cherche_dicho(x,L1))

## Q2
def cherche_dicho(e,L):
    # cette fonction retourne l'indice de l'élément e s'il est présent dans la liste triée L, No
    b1 = 0
    b2 = len(L)

    while b1 < b2:
        ind = (b1 + b2) // 2
        if e == L[ind]:
            return ind
        elif e < L[ind]:
            b2 = ind
        else:
            b1 = ind + 1
    return None

# Version recursive :
def cherche_dicho_rec(e,L,b1,b2):
    if b1 == b2:
        return None
    ind = (b1 + b2) // 2
    if e == L[ind]:
        return ind
    elif e < L[ind]:
        return cherche_dicho_rec(e,L,b1,ind)
    else:
        return cherche_dicho_rec(e,L,ind+1,b2)

L1 = [1,3,4,5,6,12,15,16]
L2 = list(range(0,12,2))

e = 1
print(cherche_dicho(e,L1))
print(cherche_dicho_rec(e,L1,0,len(L1)))

## Q3
def cherche_dicho(e,L):
    # cette fonction retourne l'indice de l'élément e s'il est présent dans la liste triée L, No
    b1 = 0
    b2 = len(L)
    iter = 0
    while b1 < b2:
        ind = (b1 + b2) // 2
        iter += 1
        if e == L[ind]:
            return ind, iter
        elif e < L[ind]:
            b2 = ind
        else:
```

```
            b1 = ind + 1
    return None, iter

L1 = [1,3,4,5,6,12,15,16]
L2 = list(range(0,12,2))
print(cherche_dicho(5,L1))
print(L2)
print(cherche_dicho(7,L2))

## Q4 Complexité dans le pire des cas
""" Le nb de tests est minimal si e est en milieu de liste, et maximal si e est en début ou en
"""

## Q5
import time
N = 1000000
L = list(range(N))

t1 = time.time()
i1,iter1 = cherche(N-1,L)
t2 = time.time()
#L.sort()
i2,iter2 = cherche_dicho(N//2-1,L)
t3 = time.time()

print('recherche linéaire : ',iter1,'iterations ,temps =', 1000*(t2-t1),'ms')
print('recherche dichotomie : ',iter2,'iterations ,temps =', 1000*(t3-t2),'ms')

## exo 3 : Résolution dichotomique de f(x)=0
# Q1 constante K°
def K(temp):
    from numpy import exp
    tabs = temp + 273.15
    return exp((92600-199*tabs)/8.31/tabs)
print('Constante K(500°C) :',K(500))

import numpy as np
import matplotlib.pyplot as plt
tabT = np.linspace(400,600,200)
tabK = K(tabT)
plt.figure('K(T)')
plt.close()
plt.plot(tabT,tabK)
#plt.grid()
plt.plot(500,7.23e-5,'or')
plt.text(490,10e-5,'7,2e-5',color='black')
plt.title("Evolution de K° avec T")
plt.show()

'''
Commentaire : La valeur à 500°C étant faible, la réaction sera peu avancée; K serait plus élevée
'''
## Q3 Résolution par fonction 'dicho'
def dicho(f, a, b, eps):
    if f(a)*f(b) > 0:
        print("Mauvais choix de l'intervalle [a,b].") ; return
    else:
        an, bn = min(a,b), max(a,b),   # initialisation de l'intervalle courant
        larg = bn - an                 # largeur courante
        cn = ( bn + an ) / 2           # resultat courant a larg/2 pres
        while larg > 2 * eps:          # resultat final au pire a eps pres
```

```
            if f(an)*f(cn) > 0:
                an = cn
            else:
                bn = cn
            cn = ( bn + an ) / 2
            larg = bn - an
    return cn
def equation(x):
    global P
    return 7.23e-5 - (16/27) * x**2 * (2-x)**2 / (1-x)**4 / P**2

P = 100 ; print('alpha à 500°C :', round(dicho(equation, 0, 0.999, 0.001),2))
# Vérif par 'bisect' :
from scipy.optimize import bisect
print('alpha à 500°C par bisect :', round(bisect(equation, 0, 0.999),2))

## Q4 Courbe alpha = f(P)
# Les pressions :
LP = [1] + [10 * k for k in range(1,21)] ; print(list(LP))
# Les valeurs de alpha :
La = []
for P in LP:
    La.append(bisect(equation, 0, 0.999))
# Le tracé :
plt.figure('alpha(P)')
plt.close()
plt.plot(LP,La)
plt.grid()
plt.title("Evolution de alpha avec P (bar), à 500°C")
plt.show()

'''
On parvient à un avancement notable grâce à un accroissement de pression non négligeable.
'''
```