Lycée Jean Moulin **Draguignan**
NSI 1re Année .....

Analyse des algorithmes

Recherche dans un tableau : Méthode brute

Voici un algorithme de recherche d'un élément dans un tableau de n valeurs:

Données : T un tableau de n valeurs
x un élément (pas forcément dans T)

fonction recherche(T,x):
$n \leftarrow$ taille(T)
Pour i allant de 0 à n-1 **faire**
  Si T[i]=x **alors**
    renvoyer True
renvoyer False

Question 1:

Expliquer pourquoi " de 0 à $n-1$"?

.....
.....
.....
.....
.....

Question 2:

Expliquer le fonctionnement de cet algorithme et en estimer la complexité dans le pire des cas

.....
.....
.....
.....
.....
.....

Exercice 1 :

Écrire la fonction `recherche(T,x)` et faites afficher le temps d'exécution avec un tableau trié de 10 000 nombres en recherchant le dernier :
Pour rappel :

```
# pour remplir une liste de 10000 nombres et la trier en place
from random import randint
L = [randint(0,100) for i in range(10000)] # pour la créer
L.sort() #pour la trier
# Pour mesurer le temps d'exécution
from timeit import default_timer as timer
debut = timer()
#code à mesurer
fin = timer()
temps = fin - debut
```

Recherche dans un tableau : Méthode - Par dichotomie

Le principe : Imaginons que nous recherchions une personne dans un annuaire classé par ordre alphabétique

- On ouvre l'annuaire au centre
- La personne est soit sur cette page, soit avant soit après
- En supposant qu'elle soit avant (on la recherche donc dans la première moitié de l'annuaire)
- On ouvre la page du milieu de cette première moitié...
- On recommence jusqu'à obtenir la bonne page ou conclure que la personne n'est pas dans l'annuaire

Question 3:

Si l'annuaire contient les noms des 7 000 000 000 de terriens. Combien de fois faut-il que je divise 7 milliards par 2 pour qu'il n'en reste qu'un ?

.....

.....

L'algorithme

Données : T un **tableau trié** de n valeurs
x l'élément à chercher (pas forcément dans T)
fonction recherche_dicho(T,x):
i ← 0
j ← l'indice du dernier élément du tableau
Tant que $i <= j$ **faire**
  Si T[(i+j)//2]=x **alors**
    renvoyer True
  Sinon
    Si T[(i+j)//2] > x **alors**
      j ← (i+j)//2-1
    Sinon
      i ← (i+j)//2+1

renvoyer False

Question 4:

Faites fonctionner cet algorithme pour ce tableau avec x=65:
T = [1, 1, 3, 10, 16, 18, 19, 22, 35, 41, 46, 52, 55, 59, 60, 65, 67, 80, 91, 100]

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Exercice 2 :

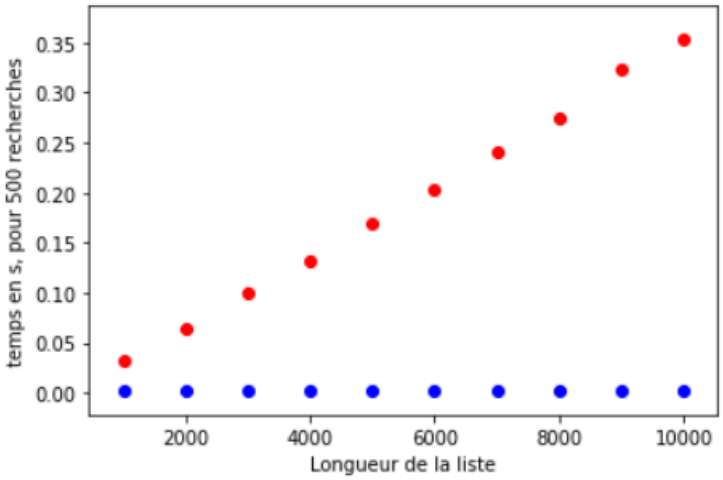
Écrire la fonction `recherche_dicho(T,x)` dans le même script que la fonction précédente et comparer leur temps d'exécution pour un même tableau trié de 10 000 nombres puis de 100 000 nombres

Afin de mettre en évidence l'intérêt de l'utilisation de la dichotomie pour une recherche dans un tableau, voici un code qui affiche un graphique montrant le temps d'exécution de 500 recherches d'un élément ne figurant pas dans des listes de longueurs 1000,2000,...,10000

```
from random import randint
from timeit import default_timer as timer
from matplotlib import pyplot as plt

abscisse=[]
ordonnee1=[]
ordonnee2=[]
for i in range(1000,11000,1000):
    L=[randint(0,10000) for k in range(i)]
    L.sort()
    abscisse.append(len(L))
    a=L[-1]+1
    s1=0
    for k in range(500):
        start=timer()
        recherche_dicho(L,a)
        end=timer()
        s1=s1+(end-start)
    ordonnee1.append(s1)

    s2=0
    for k in range(500):
        start2=timer()
        recherche(L,a)
        end2=timer()
        s2=s2+(end2-start2)
    ordonnee2.append(s2)

axes = plt.gca()
plt.xlabel('Longueur de la liste')
plt.ylabel('temps en s, pour 500 recherches')
plt.scatter(abscisse,ordonnee2,color='red')
plt.scatter(abscisse,ordonnee1,color='blue')
plt.show()
```

À faire 1:

Faites fonctionner ce code.
Vous devriez obtenir un graphique similaire à celui-ci.
Obtenu dans spyder.



Récréation

Concevoir un programme qui illustre la situation suivante :

Stéphane (S) propose à Patrice (P) le jeu suivant :

"choisis en secret un nombre compris entre 0 et 100; je vais essayer de le deviner le plus rapidement possible, en ne pouvant que te poser des questions auxquelles tu réponds par oui ou par non ".

Deux cas sont envisageables :

- Stéphane dans le rôle de l'ordinateur
- Patrice dans le rôle de l'ordinateur

Question 5:

La situation envisagée :

.....

Question 6:

Écrire l'algorithme :

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....