

IngeSUP - TD 12 - Recherche dans une liste et complexité

“ Qui cherche trouve, qui dort rêve. ”

Proverbe

Exercice 11.1

Ecrivez une fonction `RechSeq()` qui effectue une recherche séquentielle de `x` parmi les éléments d'une liste `L`. Cette fonction renverra une variable booléenne qui prendra la valeur `True` si l'élément est trouvé dans la liste ou `False` dans le cas contraire.

Entrée [17]:

```
def RechSeq(L,x):  
    for i in L:  
        if i==x:  
            return True  
    return False
```

```
RechSeq([1,4,7,8],7)
```

Out[17]:

True

Entrée [18]:

```
def RechSeq(L,x):  
    for i in range(len(L)):  
        if L[i]==x:  
            return True  
    return False
```

```
RechSeq([1,4,7,8],6)
```

Out[18]:

False

Exercice 11.2

1. Écrivez une fonction `maxi()` prenant en argument une liste d'entiers naturels `L` et renvoyant le maximum des entiers de cette liste (on n'utilisera pas de fonction spécifique de Python déterminant ce maximum).

Entrée [16]:

```
def maxi(L):  
    maxi=L[0]  
    for el in L:  
        if el > maxi:  
            maxi=el  
    return maxi  
  
# On teste la fonction sur la liste [1,7,17,11,4,2]  
print(maxi([1,7,17,11,4,2]))
```

Entrée [19]:

```
def maxi2(L):
    maxi = L[0]
    for i in range(len(L)):
        if L[i] > maxi:
            maxi = L[i]
    return maxi

# On teste la fonction sur la liste [1,7,17,11,4,2]
print(maxi2([1,7,17,11,4,2]))
```

17

2. Quelle est le nombre d'opérations élémentaires effectué par cette fonction en fonction de la longueur n de la liste?

Réponse: $O(n)$

Exercice 11.3

Écrivez une fonction `Minus()` prenant en argument une liste `Liste` et une variable `x`, et qui retourne le plus petit indice `k` de la liste tel que `Liste[k]` soit égal à `x`. Si la liste ne contient pas `x`, alors la fonction doit retourner `-1`.

Entrée [23]:

```
def Minus(L,x):
    for i in range(len(L)):
        if L[i]==x:
            return i
    return -1

print(Minus([1,4,7,8],6))
print(Minus([1,4,7,8],8))
```

-1
3

Exercice 11.4

Écrivez une fonction `Ins()` qui prend en argument une liste triée `l` et un entier `elt` et qui renvoie la liste triée obtenue par insertion à sa place de `elt` dans `l`. On fera attention à ce que la liste `l` puisse être vide.

Petite parenthèse: En python il existe une fonction pour insérer un élément à la place qu'on veut dans la liste. C'est la fonction `insert`.

Exemple:

```
aList = [123, 'xyz', 'zara', 'abc']
aList.insert( 3, 2009)
print("Final List : ", aList)
```

Va afficher *Final List : [123, 'xyz', 'zara', 2009, 'abc']*

Entrée []:

```
# Testez si vous ne me croyez pas !

aList = [123, 'xyz', 'zara', 'abc']
aList.insert( 3, 2009)
print("Final List : ", aList)
```

Entrée [49]:

```
# VOTRE CODE ICI
def ins(l,elt):
    for i in range(len(l)):
        if l[i]>elt:
            l.insert(i,elt)
            return l
    l.append(elt)
    return l

print(ins([1,2,3,6,8],5))
print(ins([1,2,3,6,8],10))
print(ins([1,2,3,6,8],7))
print(ins([1,2,3,6,8],0))
```

```
[1, 2, 3, 5, 6, 8]
[1, 2, 3, 6, 8, 10]
[1, 2, 3, 6, 7, 8]
[0, 1, 2, 3, 6, 8]
```

Exercice 11.5

1. Ecrivez une fonction de recherche itérative par dichotomie `Rech_dich_iter()` appliqué à la recherche d'une valeur `element` dans une liste `liste_triee`.

Entrée [51]:

```
def Rech_dich_iter(L, x):
    g = 0
    d = len(L) - 1
    while g <= d:
        m = (g + d) // 2
        if L[m] == x:
            return True
        if x < L[m]:
            d = m - 1
        else:
            g = m + 1
    return False

# Testez pour chercher 13 dans la Liste [1, 3, 5, 7, 8, 10, 13, 14, 17, 19]
print(Rech_dich_iter([1, 3, 5, 7, 8, 10, 13, 14, 17, 19],13))
```

True

2. Ecrivez une fonction de recherche récursive par dichotomie `Rech_dich_rec()` appliqué à la recherche d'une valeur `element` dans une liste `liste_triee`.

Entrée [54]:

```
def rech_dich_rec(L, x):
    if L[len(L)//2] == x:
        return True
    elif len(L)<2:
        return False
    else:
        if L[len(L)//2]<x :
            L=L[len(L)//2:]
            return rech_dich_rec(L, x)
        else:
            L=L[0:len(L)//2]
            return rech_dich_rec(L, x)

# Testez pour la recherche de 5
# Testez pour la recherche de 8
L=[1, 3, 5, 7, 8, 10, 13, 14, 17, 19]
print(rech_dich_rec(L, 5))
print(rech_dich_rec(L, 8))
```

True
True

Exercice 11.6

Calculez la complexité de l'algorithme suivant :

```
for i in range (m) :  
    for j in range (n) :  
        print(j-n)
```

Réponse: $O(n*m)$ ou si on veut une seule variable, soit $t=\max(n,m)$, $O(t^2)$ (en majorant le plus petit des deux par le plus grand)

Corrigé du TD 12

Vous pouvez retrouver le corrigé de ce TD [ici \(Corrig%C3%A9s/Corrig%C3%A9_TD%2012.ipynb\)](#).