TD noté 2005

L'exercice devra être imprimé à la fin du TD et rendu au chargé de TD. Il ne faut pas oublier de mentionner son nom en commentaires au début du programme.

L'objectif de cet exercice est de programmer une recherche dans une liste triée.

1) Il faut d'abord récupérer un fichier texte disponible sur Intranet [1]. Ce fichier contient un mot par ligne. Il faut lire ce fichier et construire une liste avec tous ces mots. (4 points)

2) Construire une fonction qui vérifie que la liste chargée à la question précédente est triée. (4 points)

3) Construire une fonction qui recherche un mot X dans la liste et qui retourne sa position ou -1 si ce mot n'y est pas. Cette fonction prend deux paramètres : la liste et le mot à chercher. Elle retourne un entier. On précise que pour savoir si deux chaînes de caractères sont égales, il faut utiliser l'opérateur ==. (4 points)

4) Quels sont les positions des mots "UN" et "DEUX" ? La réponse doit figurer en commentaire dans le programme. Il faudra écrire aussi le nombre de comparaisons effectuées pour trouver ces deux positions. (2 points)

5) Lorsqu'une liste est triée, rechercher un élément est beaucoup plus rapide. Si on cherche le mot X dans la liste, il suffit de le comparer au mot du milieu pour savoir si ce mot est situé dans la partie basse (X inférieur au mot du milieu), la partie haute (X supérieur au mot du milieu). S'il est égal, le mot a été trouvé. Si le mot n'a pas été trouvé, on recommence avec la sous-liste inférieure ou supérieure selon les cas jusqu'à ce qu'on ait trouvé le mot ou qu'on soit sûr que le mot cherché n'y est pas.

Le résultat de la recherche est la position du mot dans la liste ou -1 si ce mot n'a pas été trouvé. Cette recherche s'appelle une recherche dichotomique.

Ecrire la fonction qui effectue la recherche dichotomique d'un mot dans une liste triée de mots. Vérifiez que les deux fonctions retournent bien les mêmes résultats. Cette fonction peut être récursive ou non. Elle prend au moins les deux mêmes paramètres que ceux de la question 3, si elle en a d'autres, il faudra leur donner une valeur par défaut. On précise que les comparaisons entre chaînes de caractères utilisent aussi les opérateurs <, ==, >. (4 points)

6) Normalement, les positions des mots "UN" et "DEUX" n'ont pas changé mais il faut de nouveau déterminer le nombre d'itérations effectuées pour trouver ces deux positions avec la recherche dichotomique. (2 points)

7) Quel est, au pire [2], le coût d'une recherche non dichotomique ? La réponse doit figurer en commentaire dans le programme. (1 point)

8) Quel est, au pire, le coût d'une recherche dichotomique ? La réponse doit figurer en commentaire dans le programme. (1 point)

Correction

```
# coding: latin-1
# question 1
def lit_fichier (file) :
    f = open (file, "r")
```

1. http://www.xavierdupre.fr/enseignement/initiation/td_note_texte.txt
2. On cherche la meilleure majoration du coût de la recherche non dichotomique en fonction de la taille n de la liste.

```
    mot = []
    for l in f :
        mot.append ( l.replace ("\n", "") )
    f.close ()
    return mot

mot = lit_fichier ("td_note_texte.txt")
print mot

# question 2
def est_trie (mot) :
    for i in range (1, len (mot)) :
        if mot [i-1] > mot [i] :
            return False
    return True

tri = est_trie (mot)
print "liste triée ", tri

# question 3
def cherche (mot, m) :
    for i in range (0, len (mot)) :
        if mot [i] == m :
            return i
    return -1

print "mot ACHATS ", cherche (mot, "ACHATS")
print "mot achats ", cherche (mot, "achats")

# question 4
un   = cherche (mot, "UN")
deux = cherche (mot, "DEUX")
print "recherche normale ", un, deux
print "nombre d'itérations", un + deux

# question 5, 6, nbun et nbdeux contiennent le nombre de comparaisons
def cherche_dicho (mot, m) :
    a = 0
    b = len (mot)-1
    nb = 0
    while a < b :
        nb += 1
        p = (a+b)/2
        if   mot [p] == m :  return p,nb
        elif mot [p] >  m :  b = p-1
        else :               a = p+1
    return -1,nb

un,nbun     = cherche_dicho (mot, "UN")
deux,nbdeux = cherche_dicho (mot, "DEUX")
print "recherche dichotomique ", un, deux
print "nombre d'itérations ", nbun + nbdeux

# question 7
"""
Lors d'une recherche simple, au pire, l'élément cherche sera
en dernière position, ce qui signifie n itérations pour le trouver.
Le coût de la recherche simple est en O(n).
"""
```

```
# question 8
"""
Lors de la recherche dichotomique, à chaque itération, on divise par deux
l'ensemble dans lequel la recherche s'effectue,
au départ n, puis n/2, puis n/4 jusqu'àă ce que n/2^k soit nul
c'est-à-dire k = partie entière de ln n / ln 2
il y a au plus k itérations donc le coût de l'algorithme est en O (ln n).
"""
```