Lycée Thiers — Informatique
Année 2014/15 — MPSI

TP 6: Feuille d'Exercices

Dichotomie pour la recherche d'un élément dans une liste ordonnée de nombres.
Dichotomie pour la recherche d'une racine de fonction.
Méthode de Newton pour la recherche d'une racine de fonction.

Exercice 1. *Dichotomie pour la recherche d'un élément dans une liste triée.*

(1) Ecrire une fonction `recherche(l,e)` prenant en paramètre un nombre `e` et une liste `l` et qui recherche si `e` apparaît ou non dans la liste `l`. La fonction retournera le booléen `True` ou `False`.
   (a) en utilisant l'instruction `in`.
   (b) sans utiliser l'instruction `in`.

(2) Ecrire une fonction `dich_search()` qui recherche par dichotomie si un élément est présent dans une liste de nombres ordonnée dans le sens croissant.
   (a) A l'aide d'un slicing.
   (b) Sans utiliser de slicing. On s'inspirera du code suivant que l'on complétera :

```
def dich_search2(l,e):
    Imin, Imax = 0, len(l)-1
    while Imax - Imin >= 0:
        Imed = ............
        if ..........:
            return True
        elif ..........:
            Imin = Imed + 1
        else:
            Imax = Imed - 1
    return ............
```

(3) Tester le temps d'exécution des 4 algorithmes de recherche sur une liste aléatoire ordonnée de 100 000 de nombres, grâce aux commandes suivantes (que l'on complétera) :

```
from random import random
L = [random() for k in range(100000)]    # Liste aléatoire
L.sort()    # Tri de la liste
from time import clock
a = clock()
recherche(L,0)
b = clock()
print(b-a, 'secondes')    # Temps d'éxécution
# et de même avec les 3 autres fonctions de recherche ...
```

Que constate-t-on ?

Exercice 2. *Dichotomie pour la recherche d'une racine de fonctions.*
Soit la fonction

$$f : x \mapsto x^3 - 3x^2 + 1$$

(on pourra la définir par : `f = lambda x:  x**3-2*x**2+1` grâce à l'instruction `lambda`.)

(1) Vérifier que $f(0) = 1 > 0$ et $f(2) = -3 < 0$, et en déduire l'existence d'une solution unique dans $[0, 2]$ à l'équation $f(x) = 0$.

(2) Ecrire une fontion `dich_solve()` prenant en paramètre une fonction f, un entier n, des réels (float) `a<b` tels que `f(a)f(b)`< 0 et qui retourne une valeur approchée à 10^{-n} près de la solution. La recherche s'effectuera *par dichotomie sur l'intervalle* `[a,b]`.

(3) Combien de passages dans la boucle `while` sont nécessaires pour obtenir une valeur approchée à 10^{-5} près ?

Exercice 3. *Une recherche de racine encore plus rapide !.*
La méthode de Newton permet souvent de déterminer efficacement une valeur approchée de la racine d'une fonction :

Théorème 1. *Si $f : I \longrightarrow \mathbb{R}$ est de classe C^1 (dérivable à dérivée continue) et a une racine r isolée sur l'intervalle ouvert I, si x_0 est suffisamment proche de r et si $f'(r) \neq 0$, alors la suite : (x_n) définie par $\forall\, n \in \mathbb{N}$, $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)}$ converge vers la racine r.*

(1) En quel point la tangente à la courbe représentative en x_n intersecte-t-elle la droite des abscisses ? En déduire une interprétation graphique de la suite (x_n).

(2) Ecrire une fonction `newton(f,f',x,n)` qui retourne la valeur approchée x_n de la racine de f. Retrouver le résultat de l'Exercice 2, en partant du point $x = 1$. Que se passe-t-il si l'on prend $x = 0$ ou 2 ?

(3) Pour la résolution de $x^n = a$, c'est à dire pour l'extraction de racines n-ièmes cette méthode s'appelle la *méthode de Héron d'Alexandrie*. L'appliquer pour le calcul approché de $\sqrt{2}$ et comparer le résultat obtenu à la valeur réelle.