Informatique PCSI
TP 4 : algorithmes dichotomiques

1 Énoncé des exercices

1.1 Exercice 1

Il s'agit de simuler un jeu où un nombre entier est choisi au hasard par la machine, entre 1 et 100, ou 1 et 1000 par exemple, et le joueur doit deviner ce nombre en un minimum de coups. À chaque étape le joueur propose un nombre, entré au clavier, et la machine répond en affichant un message disant si le nombre est trop grand, trop petit, ou si le joueur a gagné.

1. Écrire une fonction `devine` qui prend en argument un entier `n` et réalise ce jeu où l'entier à deviner est compris entre 1 et `n`

   Quelle est la stratégie optimale permettant de gagner avec le minimum de coups ?
2. Améliorer le jeu en affichant à chaque étape le nombre de coups joués, ou le nombre de coups qui restent à jouer si un nombre de coups maximal a été défini.

Voici deux instructions utiles pour choisir un nombre au hasard et pour enregistrer un nombre entré au clavier :

```
from random import randint
nombre = randint(1, n)
# n est défini préalablement

# choix du joueur
choix = int(input("Entrer un nombre: "))
```

1.2 Exercice 2

Tester puis corriger la fonction suivante qui doit renvoyer une solution approchée sur un intervalle $[a;b]$ de l'équation $f(x) = 0$. La fonction f est supposée continue monotone et a une solution unique sur $[a;b]$.

```
def dicho(f, a, b, p):
    while b-a > p:
        m = (a + b) / 2
        if f(a) * f(m) < 0:
            b = m
        else:
            a = m
    return (a + b) / 2
```

Utiliser des fonctions simples pour les tests, comme f et g définies respectivement par $f(x) = x$ et $g(x) = x^2$.

1.3 Exercice 3

On utilise la première version de la fonction `dichotomie` présentée dans le document préparatoire.

1. On recherche dans la liste `[2, 4, 6]` la valeur 6 puis la valeur 7 à l'aide de cette fonction. Expliciter dans chaque cas les valeurs successives des variables `g`, `d` et `k`.
2. Peut-on avoir une des valeurs successives de `liste[d]` égale à la valeur cherchée `x` ?

1.4 Exercice 4

Une recherche est effectuée avec la deuxième version de la fonction `dichotomie` du document préparatoire, dans la liste `[12, 15, 16, 18, 21, 24, 25, 27, 31, 33, 35, 36, 38]`.

Donner l'ordre de parcours des différentes valeurs dans une recherche dichotomique.

1.5 Exercice 5

On étudie le code de la deuxième version de la fonction `dichotomie` citée ci-dessus. Il s'agit d'expliciter certains détails de ce programme.

1. Quel est le problème si on remplace la condition `g <= d` par la condition `g < d` ?
2. Quel est le problème si on remplace l'affectation `g = m+1` par l'affectation `g = m` ?
3. Quel est le problème si on remplace l'affectation `d = m-1` par l'affectation `d = m` ?
4. Lorsque la condition `g <= d` n'est plus satisfaite, quelle relation peut-on écrire entre `g` et `d` ? Que peut-on dire alors de `liste[g]`, `x` et `liste[d]` ?

Pour les questions 1, 2 et 3, on peut considérer une liste à deux éléments comme `[2, 4]` et la recherche de la valeur 4 pour les questions 1 et 2, celle de la valeur 3 pour la question 3.

Pour la question 4, examiner séparément les cas qui conduisent à une situation `g > d`.

1.6 Exercice 6

Écrire une fonction `insertion` qui prend en paramètres une liste de nombres triée et un nombre x et insère le nombre x dans la liste de manière à ce que la liste reste triée. La fonction utilise un algorithme dichotomique pour trouver la bonne place et ne renvoie rien.

Utiliser un code semblable à celui d'une recherche dichotomique.