Feuille d'Exercices n°9
Dichotomie

Dans ce TP, on s'intéresse à différentes applications de la technique de *dichotomie*. Cette technique est utile pour rechercher un élément dans un espace de recherche (par exemple une liste). Le principe de la recherche dichotomique est de couper l'espace de recherche en deux, successivement jusqu'à trouver l'élément. C'est une technique beaucoup plus efficace que de parcourir l'ensemble des éléments de l'espace, un à un.

Exemple 1. Dans 'Le Juste Prix', les candidat.es finalistes ont 30 secondes pour deviner à l'euro près la valeur d'une vitrine de cadeaux (et remportent les cadeaux s'iels y arrivent). La vitrine a une valeur maximale de 50.000 euros. Pour ce faire, le/la candidat propose une estimation, et l'animateur lui répond "c'est plus" ou "c'est moins".
La technique optimale consiste à proposer successivement le milieu des bornes du prix. Ainsi :

— On commence par proposer 25.000€

— Si c'est plus, on propose 37500€, si c'est moins 12500€

— Et on continue comme ça

Dans le pire des cas, il faudra 16 essais pour trouver le juste prix.
Pourquoi 16 ? Et si la valeur maximum de la vitrine est 1.000.000€, combien d'essais faudra-t-il au maximum ?

Exercice 1 *Recherche dichotomique dans une liste triée*

Dans cet exercice on se pose la question d'écrire un algorithme pour déterminer si un élément appartient à une liste.

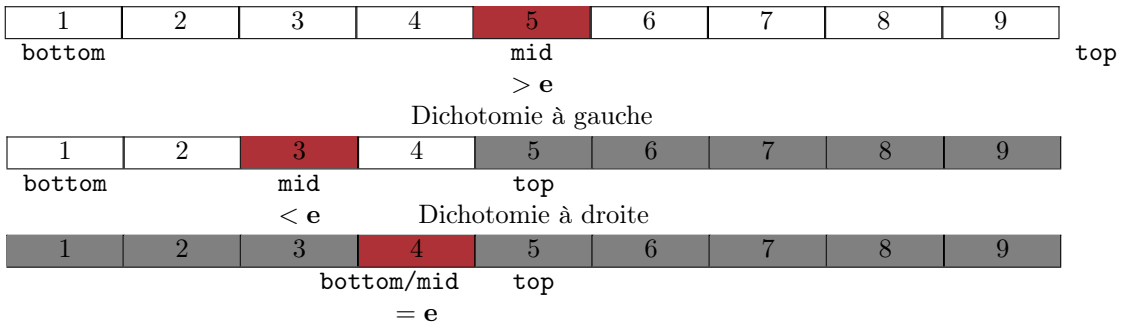
1. a) Écrire une fonction `find_naive(l, x)` qui prend en argument une liste `l1` et un élément `x` et détermine si `x` apparaît dans `L`. La fonction doit renvoyer `True` ou `False` en fonction.

b) Tester votre fonction sur quelques exemples.

c) Si `L` est de taille 10, combien d'éléments est-ce que `find_naive` devra regarder dans le pire des cas ? Et si `L` est de taille 1000 ?

Dans le cas général, on ne peut pas vraiment faire mieux que de regarder un à un tous les éléments. Mais si `t` est triée (dans l'ordre croissant), on peut aller beaucoup plus rapidement ! En effet si je sais que `x` est plus grand qu'un élément `e` de la liste, alors il ne peut pas être à gauche de `e`. S'il est plus petit, `x` ne peut pas être à droite de `e`.

Exemple 2. Recherche de 4 dans le tableau suivant :



2. a) Compléter la fonction `find_dichotomique(l, x)` qui prend en argument une liste `L` *triée* et un élément `x` et détermine si `x` apparaît dans `L`.

```
def find_dichotomique(L, x):
    """Returns True if x is in L and False otherwise"""
    bottom, top = 0, len(L)
    while top - bottom > 0:
        mid = ...
        y = L[mid]
        if x == y:
            ...
        elif x > y:
            ...
        elif x < y:
            ...
    return False
```

b) Tester votre fonction sur quelques exemples (vérifier qu'elle fonctionne avec la liste vide)

c) Si `L` est de taille 10, combien d'éléments est-ce que `find_dichotomique` devra regarder dans le pire des cas ? Et si `L` est de taille 1000 ?

d) Que se passe-t-il si `L` n'est pas triée ?

e) Modifier la fonction pour prendre en argument une liste triée dans l'ordre décroissant.

3. On va mettre en évidence la différence de performance de `find_dichotomique` versus `find_naive`.

a) Utiliser le code suivant pour mesurer le temps de calcul des fonctions.

```
from random import random
from time import time

L = [random() for k in range(10_000)]  # a list of random floats between 0 and 1
L.sort()

start = time()
find_naive(L, 0.5)
end = time()
print("find_naive took {} seconds".format(end - start))

start = time()
find_dichotomique(L, 0.5)
end = time()
print("find_dichotomique took {} seconds".format(end - start))
```

b) Tester sur des listes de taille 10^6. Que remarquez-vous ?

Exercice 2 *Recherche dichotomique d'un zéro d'une fonction*

Dans cet exercice, on utilise la dichotomie pour trouver un point d'annulation d'une fonction *continue*. On rappel le théorème des valeurs intermédiaires :

Théorème 3. *Soit $f : [a, b] \to \mathbb{R}$ continue. Si $f(a) \cdot f(b) \leqslant 0$, alors il existe $c \in [a, b]$ tel que $f(c) = 0$.*

Supposez qu'on vous fournisse une fonction continue f et $a < b$ tels que $f(a) < 0$ et $f(b) > 0$. On sait qu'il existe quelque part entre a et b un zéro de f. Peut-on trouver un intervalle plus petit contenant un zéro de f ? Oui par une méthode dichotomique ! On regarde le signe de $f(\frac{a+b}{2})$.

— S'il est négatif, alors il existe un zéro de f entre $\frac{a+b}{2}$ et b

— S'il est positif, alors il existe un zéro de f entre a et $\frac{a+b}{2}$

On a réduit la taille de l'intervalle par 2 !

1. a) Compléter le code suivant qui calcule un intervalle contenante un zéro de la fonction $x \mapsto \sin(x)$.

```
from math import sin

def find_zero(a, b, n_iteration):
    if sin(a) > 0 or sin(b) < 0:
        return "Cannot do a dichotomy"

    for i in range(n_iteration):
        mid = (a + b)/2
        if sin(mid) > 0:
            a, b = ...
        else:
            a, b = ...

    return a, b
```

b) Quel est la taille de l'intervalle renvoyé par `find_zero(a, b, n_iteration)`, en fonction de `a`, `b` et `n_iteration`

c) Comment utiliser le code précèdent pour obtenir une approximation de π au millième près ?

2. Modifier la fonction `find_zero` pour qu'elle fonctionne également dans le cas où $\sin(a) > 0$ et $\sin(b) < 0$.