ISM MPSI/PCSI, Informatique
Année 2025/2026

Corrigé du TP Informatique 11

Exercice 1

On saisit :

```
def rech1(elt,L):
    """rech1(elt:int,L:list)->(bool,int)
    Renvoie le résultat de la recherche dichotomique
    de elt dans L liste triée :
    * si L[k]==elt       -> (True,k)
    * si elt absent de L -> (False,k)"""
    deb=0
    fin=len(L)-1
    trouve=False
    while not trouve and deb<=fin:
        milieu=(deb+fin)//2
        #print(deb,fin,L[deb:fin+1])
        if L[milieu]==elt:
            trouve=True
        elif L[milieu]>elt:
            fin=milieu-1
        else:
            deb=milieu+1
    return L[milieu]==elt,milieu
```

Exercice 2

1. On constate sans difficulté que l'instruction `sorted` renvoie une liste ordonnée :

```
>>> sorted([3,1,4,2,0])
[0, 1, 2, 3, 4]
```

2. On saisit :

```
def dmin_sort(L):
    """Distance minimale entre éléments de L"""
    tab=sorted(L)
    n=len(L)
    res=tab[1]-tab[0]
    for k in range(n-1):
        dcur=tab[k+1]-tab[k]
        if dcur<res:
            res=dcur
    return res
```

3. On saisit :

```
def dmin(L):
    """Distance minimale entre éléments de L"""
    dmin=abs(L[0]-L[1])
    for i in range(1,len(L)):
        for j in range(i):
            dcur=abs(L[i]-L[j])
            if dcur<dmin:
                dmin=dcur
    return dmin
```

4. On teste :

```
import numpy.random as rd
L=rd.rand(10000)
print(dmin_sort(L))
print(dmin(L))
```

On obtient les mêmes résultats :

```
4.60855797968e-09
4.60855797968e-09
```

ce qui est normal mais surtout, on constate une très grande différence de temps de calcul : presque instantané pour `dmin_sort`, vraiment long pour `dmin`. On décryptera ce phénomène dans le chapitre **Complexité**.

Exercice 3

3. On saisit :

```
deb=0
fin=1000

a=0
b=0
for k in range(deb,fin):
    a+=1
    b+=1
    if id(a)!=id(b):
        print(k)
        break
```

ou

```
deb=0
fin=1000

a=0
b=0

while id(a)==id(b):
    a+=1
    b+=1
print(a)
```

4. On saisit :

```
deb=0
fin=1000


while deb<fin:
    m=(deb+fin)//2
    a=m
    m=(deb+fin)//2
    b=m
    #print(m)
    if id(a)==id(b):
        deb=m+1
    else:
        fin=m-1

print(deb)
```

Le phénomène mis en avant par les différents tests et programmes est le suivant : les entiers jusqu'à 256, plus précisément dans la plage $[\![-5 ; 256]\!]$ ont un emplacement dédié en mémoire. En dehors de cette plage, python alloue une certaine zone de mémoire pour l'écriture d'un entier et celle-ci n'est pas à adresse fixe.