01.03.2021 (présentiel)

EA TD 4 Dichotomie et tri

Exercice 1

1)
```
occurencesNaif(T, x):
    nb = 0
    for el in T:                 n tours de boucle
        if el == x:              } Θ(1)
            nb += 1
    return nb
```

Complexité : $\Theta(n)$

2)
```
occurencesDicho(T, x):
    i = findMin(T, x)            Θ(log(n))
    j = findMax(T, x)            Θ(log(n))
    return j - i                 Θ(1)
```

Complexité : $\Theta(\log(n))$

```
def findMax(T, x, i, j):
    if j - i == 1:
        return i
    k = (i+j)/2
    if T[k] > x:
        return findMax(T, x, i, k)
    else if T[k] <= x
        return findMax(T, x, k+1, j)
```

```
def findMin(T, x, i, j):
    if j - i == 1:
        return i
    k = (i+j)/2
    if T[k] >= x:
        return findMin(T, x, i, k)
    else if T[k] < x
        return findMin(T, x, k+1, j)
```

Exercice 2

1)
```
compteNaif(T):
    m = max(T)                   → Θ(n)
    S = [0] * m
    for i in range(n):           → n tours de boucle
        if S[i] == 0
            S[i] = occurencesNaif(T, i)   → Θ(n)
    return S
```

Complexité :

mémoire : $\Theta(n)$

temps : $\Theta(n) + \Theta(m \times n) = \Theta(m \times n)$

La taille de S est d'au moins l'élément max de T.

2)
```
compteOptimal(T):
    S = [0] * max(T)             Θ(n)
    for el in T:                 n tours de boucle
        S[el] += 1               Θ(1)
    return S
```

Complexité en tps : $\Theta(n)$

3)
```
isPermutation(T):
    S = compteOptimal(T)
    if len(S) != len(T) + 1:
        return false
    for el in S:
        if (el != 1) return false
    return true
```

Exercice 3

1)

PIRE	Gauche	Droite
comparaisons	$\frac{n(n-1)}{2}$	
échanges		
les 2	$\Theta(n^2)$	$\Theta(n^2)$

MEILLEUR	Gauche	Droite
comparaisons	$n-1$	$n-1$
échanges	0	0
les 2	$n-1$	$n-1$

Les deux algos ont la même complexité.

2) Init : Avant la première itération, $i = 0$
Le tableau $T[:1]$ contient un seul élément donc il est trié.

Conservation : Pour chaque tour de boucle i, le tableau $T[:i]$ est trié par hypothèse de récurrence.
On décale les éléments de $T[:i]$ vers la droite de manière à insérer $T[i]$ à la bonne place.
On a alors un tableau $T[:i+1]$ trié.

Terminaison : À la fin de la boucle, $i = \text{len}(T) - 1$.
Donc $T[:\text{len}(T)] = T$ est trié.

3) L'échange élimine l'inversion entre $T[j]$ et $T[j-1]$.
Si $T[j]$ est arrivé à la bonne place dans le tableau, alors le nb d'inversions diminue de 1.
Sinon, il y a une nouvelle inversion entre $T[j-1]$ et $T[j-2]$
↑ l'ancien $T[j]$.

4) Un échange par inversion, c'est 3 affectations.
L'algo fait dans le pire des cas, $(n-1) \times \frac{n(n-1)}{2} = \frac{2(n-1)n}{2} = n(n-1)$ échanges par inversion.

En tout ça donne $3n(n-1)$ affectations.

5) On sauvegarde la valeur de $T[j]$, on décale les éléments inférieurs (1 affectation au lieu de 3) puis on insère la valeur sauvegardée.

6) Dans le pire des cas, l'algo fait $(n-1) \times \frac{n(n-1)}{2} = n(n-1)$ comparaisons.

7) On peut sortir du for intérieur dès que la comparaison échoue car le reste du tableau est trié.

8)