ISM MP, Informatique
Année 2025/2026

Corrigé du TP Informatique 03

Exercice 1

1. On saisit :

```
def tri_ins(T):
    n=len(T)
    for i in range(1,n):
        j=i
        while j>0 and T[j-1]>T[j]:
            T[j],T[j-1]=T[j-1],T[j]
            j-=1
```

2. L'implémentation effectuée modifie directement le liste à trier et on ne crée aucune variable de taille de même ordre que celle de l'argument `T` ce qui prouve le caractère en place de ce tri. On considère une liste de taille n. Si celle-ci est déjà triée, on ne rentre jamais dans la boucle `while` et le coût temporel est en

$$\sum_{i=1}^{n-1} \mathrm{O}(1) = \mathrm{O}(n)$$

Dans le pire des cas, on rentre `i` fois dans la boucle `while` d'où un coût en

$$\sum_{i=1}^{n-1}\sum_{j=1}^{i} \mathrm{O}(1) = \sum_{i=1}^{n-1} \mathrm{O}(i) = \mathrm{O}\left(\sum_{i=1}^{n-1} i\right) = \mathrm{O}(n^2)$$

> Le tri par insertion est en place de complexité temporelle en $\mathrm{O}(n)$ dans le meilleur des cas et en $\mathrm{O}(n^2)$ dans le pire des cas.

3. On saisit :

```
def tri_ins(T):
    n=len(T)
    for i in range(1,n):
        c=T[i]
        j=i-1
        while j>=0 and T[j]>c:
            T[j+1]=T[j]
            j-=1
        T[j+1]=c
```

Cette amélioration ne change pas la classe de complexité.

Exercice 2

On saisit :

```
def tri_rapide(T):
    if T==[]:
        return []
    else:
        pivot=T[0]
        T1,T2=[],[]
        for x in T[1:]:
            if x<pivot:
                T1.append(x)
            else:
                T2.append(x)
        return tri_rapide(T1)+[pivot]+tri_rapide(T2)
```

Exercice 3

1. On saisit :

```
def fusion(T,g,m,d):
    tab=T[g:m]+T[d-1:m-1:-1]
    i=0
    j=-1
    for k in range(g,d):
        if tab[i]<tab[j]:
            T[k]=tab[i]
            i+=1
        else:
            T[k]=tab[j]
            j-=1
```

2. On saisit :

```
def tri_fusion_rec(T,g,d):
    if d-g>1:
        m=(g+d)//2
        tri_fusion_rec(T,g,m)
        tri_fusion_rec(T,m,d)
        fusion(T,g,m,d)
```

3. On saisit :

```
def tri_fusion(T):
    tri_fusion_rec(T,0,len(T))
```

4. Il ne s'agit pas d'un tri en place car la variable `tab` créée par la fonction `fusion` est de même taille que l'argument d'entrée après les derniers appels récursifs.