SÉRIE N° 4 : COMPLEXITÉ ALGORITHMIQUE (SUITE)

Exercice 7: Donner la complexité dans le pire et le meilleur des cas de la fonction suivante :

```
def Somme (M):
        n= len(M)
        s=0
        for i in range(n):
                for j in range(n):
                        s+=M[i][j]
        return s
```

Exercice 8: Donner la complexité dans le pire et le meilleur des cas de la fonction suivante :

```
def Triangulaire (M):
        n= len(M)
        for i in range(n):
                for j in range(i+1):
                        print(M[i][j])
```

Exercice 9:

1. Ecrire une fonction **sansDoublons(t)** qui retourne True si le tableau d'entiers t est sans doublons (c'est à dire sans apparition multiple d'un élément), False sinon.

   Exemples :

   ```
   >>> a = [1,4,3]
   >>> b = [1,0,0]
   >>> sansDoublons(a) : True
   >>> sansDoublons(b) : False
   ```

2. Donner et justifier la complexité dans le pire et le meilleur des cas de cette fonction.

Exercice 10:

1. Ecrire la fonction **Produit(A,B)** qui permet de retourner le produit matriciel de deux matrices carrées A et B.
2. Donner la complexité dans le pire et le meilleur des cas de la fonction

Exercice 11: Donner la complexité pour chaque Tri :

Tri	Complexité

def tri_selection(L):
for i in range(len(L)-1):
k=i
for j in range(i+1,len(L)):
if L[j]<L[k]:
k=j
L[i],L[k]=L[k],L[i]	

def tri_bulle(L):
n = len(L)
for j in range(n-1,0,-1):
for i in range(j):
if L[i] > L[i+1]:
L[i],L[i+1]=L[i+1],L[i]	