TD 4 : ASD

Exercice 3 :

```python
def algo(data):
    n = len(data)
    i = 0
    swapped = True
    while i < n-1 and swapped:
        swapped = False
        for j in range(n-i-1):
            if data[j] > data[j+1]:
                data[j], data[j+1] = data[j+1], data[j]
                swapped = True
        i += 1
```

Q 3.1 Déroulez le fonctionnement de l'algorithme pour `data = [7, 3, 5, 2]`, en indiquant pour chaque changement de `data`, les valeurs de `i` et `j`, ainsi que sa nouvelle valeur.

Q 3.2 De manière générale, exprimez ce que serait un pire des cas et un meilleur des cas pour cet algorithme.

Q 3.3 Dans le pire des cas, quelle est la complexité de l'algorithme ?

data	i	j	data_n
7,3,5,2	0	0	"
"	"	1	3,7,5,2
3,7,5,2	"	2	"
"	"	3	2,7,5,3
2,7,5,3	1	1	"
"	1	2	2,5,7,3
2,5,7,3	1	3	2,3,7,5
2,3,7,5	2	2	"
"	"	3	2,3,5,7

Meill ^R cas : liste triée

Pire cas : liste triée décroissante

Q3.3 $\sum_{i=0}^{n-2} \sum_{k=0}^{n-i-1} 1 = \sum_{i=0}^{n-2} (n-i-1) = \sum_{i=0}^{n-2} (n) - \sum_{i=0}^{n-2} i - \sum_{i=0}^{n-2} 1$

$= (n-1)n - \frac{(n-1)(n-2)}{2} - (n-2)$

$= (n-1)\left(\frac{-(n-2)+2n}{2}\right) - (n-2)$

$= O(n^2)$

Exercice 5 :

```python
"""
  CU: there exists p such that n = 2^p
"""
def exo4 (n):
    r = 0
    while n != 0:
        n = n // 2
        r = r + 1
    return r
```

Q 5.1 Calculer le nombre d'itérations effectuées en fonction de n.

$\log(n)$

Exercice 6 : Table redimensionnable

On considère une table avec redimensionnement dynamique : dès que la table a été intégralement remplie, on crée une nouvelle table de taille double et on y recopie toutes les éléments de la table précédente.

Imaginons qu'on réalise au total n insertions dans une table dont la taille était initialement 2^0.

Q 6.1 Montrer que le coût en temps est de l'ordre de n malgré les redimensionnements et recopies nécessaires.

Q 6.2 En déduire qu'en moyenne la complexité de l'insertion est en $\mathcal{O}(1)$ pour les tables avec redimensionnement dynamique.

Pour une liste de taille n, on fait $1 + 2 + 4 + \dots + \log_2(n)$

$\sum_{i=0}^{\log_2(n)-n} 2^i = \frac{1 - 2^{\log_2(n)}}{1-2} = \frac{1+n}{+1} = \Theta(n)$