Tri à Bulles

Principe

Le tri à bulles consiste à comparer répétitivement les éléments consécutifs d'un tableau. Si une paire d'éléments consécutifs est mal ordonnée, on en permute les éléments.

Il doit son nom au fait qu'il déplace rapidement les plus grands éléments en fin de tableau, comme des bulles d'air qui remonteraient rapidement à la surface d'un liquide.

Le coeur de l'algorithme consiste à échanger deux éléments consécutifs si le second est strictement plus grand que le premier

In [1]:
```
def echangeSiNecessaire( A, i):
    # Après cette fonction, A[i] <= A[i+1]
    if A[i] > A[i+1]:
        A[i], A[i+1] = A[i+1], A[i]    # swap
```

Appliquons le par exemple aux troisièmes et quatrièmes éléments (d'indices 2 et 3)

In [2]:
```
T = [ 5, 8, 4, 3, 2, 6, 7, 1 ]; print(T)
echangeSiNecessaire(T,2);       print(T)
echangeSiNecessaire(T,2);       print(T)
```

```
[5, 8, 4, 3, 2, 6, 7, 1]
[5, 8, 3, 4, 2, 6, 7, 1]
[5, 8, 3, 4, 2, 6, 7, 1]
```

Boucle interne

La boucle interne du tri à bulles passe sur toutes les paires d'éléments et teste s'il est nécessaire de les échanger

In [3]:
```
def boucleInterne(A):
    N = len(A)
    for i in range(0,N-1):
        echangeSiNecessaire(A,i)
        print(A)
    return;
```

In [4]:
```
T = [ 4, 5, 8, 3, 2, 6, 7, 1 ]
boucleInterne(T)
```

```
[4, 5, 8, 3, 2, 6, 7, 1]
[4, 5, 8, 3, 2, 6, 7, 1]
[4, 5, 3, 8, 2, 6, 7, 1]
[4, 5, 3, 2, 8, 6, 7, 1]
[4, 5, 3, 2, 6, 8, 7, 1]
[4, 5, 3, 2, 6, 7, 8, 1]
[4, 5, 3, 2, 6, 7, 1, 8]
```

Notons qu'à la fin de cette boucle, le plus grand élément du tableau est à sa place finale, tout à droite. Un nouvel appel à cette boucle effectue donc un test inutile entre les deux élément de la dernière paire.

In [5]:
```
boucleInterne(T)
```

```
[4, 5, 3, 2, 6, 7, 1, 8]
[4, 3, 5, 2, 6, 7, 1, 8]
[4, 3, 2, 5, 6, 7, 1, 8]
[4, 3, 2, 5, 6, 7, 1, 8]
[4, 3, 2, 5, 6, 7, 1, 8]
[4, 3, 2, 5, 6, 1, 7, 8]
[4, 3, 2, 5, 6, 1, 7, 8]
```

De même, après ce second passage les deux éléments les plus grands sont maintenant en place. Il est donc plus pertinent de limiter l'effet de la boucle interne aux k premier éléments de tableau (k-1 paires)

In [6]:
```
def boucleInterne(A,k):
    for i in range(0,k-1): # paires (0,1), (1,2), ... (k-2,k-1)
        echangeSiNecessaire(A,i)
        print(A[0:k],A[k:len(A)])
    return;
```

In [7]:
```
boucleInterne(T,6)
```

```
[3, 4, 2, 5, 6, 1] [7, 8]
[3, 2, 4, 5, 6, 1] [7, 8]
[3, 2, 4, 5, 6, 1] [7, 8]
[3, 2, 4, 5, 6, 1] [7, 8]
[3, 2, 4, 5, 1, 6] [7, 8]
```

In [8]:
```
boucleInterne(T,5)
```

```
[2, 3, 4, 5, 1] [6, 7, 8]
[2, 3, 4, 5, 1] [6, 7, 8]
[2, 3, 4, 5, 1] [6, 7, 8]
[2, 3, 4, 1, 5] [6, 7, 8]
```

In [9]:
```
boucleInterne(T,4)
```

```
[2, 3, 4, 1] [5, 6, 7, 8]
[2, 3, 4, 1] [5, 6, 7, 8]
[2, 3, 1, 4] [5, 6, 7, 8]
```

In [10]:
```
boucleInterne(T,3)
```

```
[2, 3, 1] [4, 5, 6, 7, 8]
[2, 1, 3] [4, 5, 6, 7, 8]
```

In [11]:
```
boucleInterne(T,2)
```

```
[1, 2] [3, 4, 5, 6, 7, 8]
```

Boucle externe

Le tri complet consiste à appeler autant de fois que nécessaire cette boucle interne. Le premier appel lui demande de passer sur les N éléments du tableau, puis N-1, N-2, ... jusqu'au dernier appel qui demande d'en traiter 2. Il est en effet inutile de trier un tableau d'un seul élément.

In [12]:
```
def boucleInterne(A,k):
    for i in range(0,k-1):
        echangeSiNecessaire(A,i)

def boucleExterne(A):
    N = len(A)
    for k in range(N,1,-1):     # N, N-1, ... 3, 2
        boucleInterne(A,k)
        print(A[0:k],A[k:len(A)])
```

In [13]:
```
T = [ 4, 5, 8, 3, 2, 6, 7, 1 ]
boucleExterne(T)
```

```
[4, 5, 3, 2, 6, 7, 1, 8] []
[4, 3, 2, 5, 6, 1, 7] [8]
[3, 2, 4, 5, 1, 6] [7, 8]
[2, 3, 4, 1, 5] [6, 7, 8]
[2, 3, 1, 4] [5, 6, 7, 8]
[2, 1, 3] [4, 5, 6, 7, 8]
[1, 2] [3, 4, 5, 6, 7, 8]
```

En résumé

Le tri à bulles effectue deux boucles imbriquées. La boucle extene s'effectue N-1 fois, la boucle interne de N-1 à 1 fois.

In [14]:
```
def tri_a_bulles(T):
    N = len(T)
    for k in range(N,1,-1):      # N, N-1, ... 2
        for i in range(0,k-1):   # 0, 1, ... k-2
            if T[i] > T[i+1]:    # T0 < T1 ... Tk-2 < Tk-1
                T[i],T[i+1] = T[i+1],T[i]   # swap
```

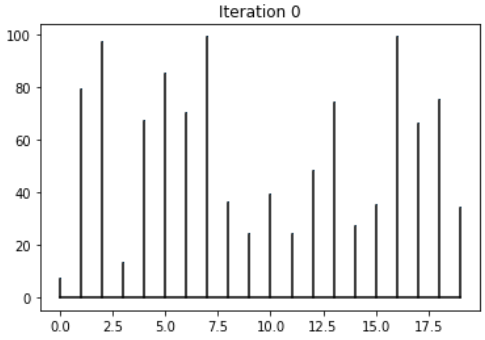
In [15]:
```
T = [5, 8, 4, 3, 2, 6, 7, 1]
tri_a_bulles(T)
print(T)
```

```
[1, 2, 3, 4, 5, 6, 7, 8]
```

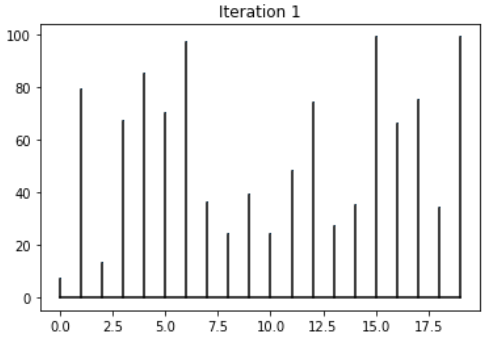
Visualisation

Pour cette visualisation, trions 20 entiers entre 0 et 100 par ordre croissant

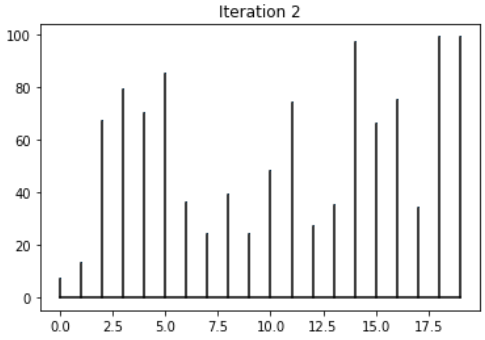
In [17]:
```
T = np.random.randint(0,100,20)
N = len(T)
it = 0
affiche(T,0)
```



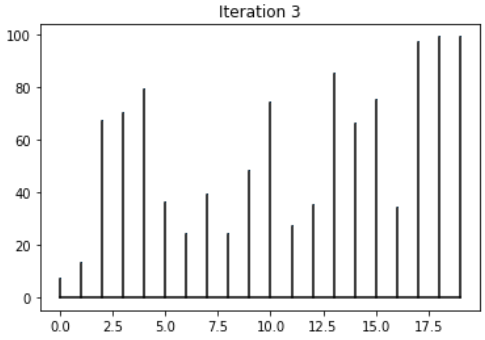
In [18]:
```
boucleInterne(T,20)
affiche(T,1)
```



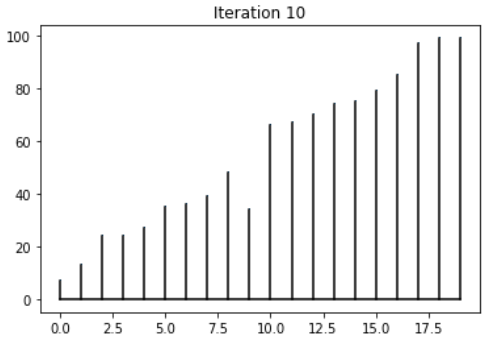
In [19]:
```
boucleInterne(T,19)
affiche(T,2)
```



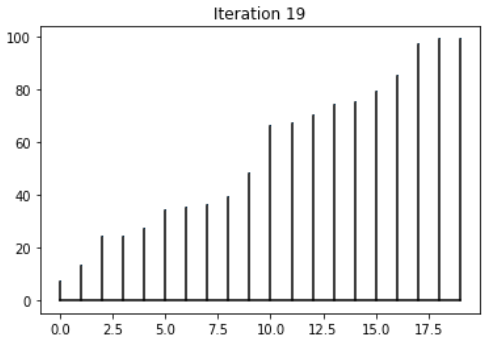
In [20]:
```
boucleInterne(T,18)
affiche(T,3)
```



In [21]:
```
for i in range(17,10,-1):
    boucleInterne(T,i)
affiche(T,10)
```



In [22]:
```
for i in range(10,1,-1):
    boucleInterne(T,i)
affiche(T,19)
```



heig-vd

ASD1 Notebooks on GitHub.io
(https://ocuisenaire.github.io/ASD1-notebooks/)