Informatique - Chapitre 1 - Bonus
Algorithmes de tris

```python
# Tri sélection
def tri_selection(l):
    for i in range(len(l)):
        minimum = l[i]
        imin = i
        for j in range(i+1,len(l)):
            if l[j] < minimum:
                minimum = l[j]
                imin = j
        if imin > i:
            l[i], l[imin] = l[imin], l[i] # permutation
        print((imin,i), l)

#============================================================================
# Tri insertion
def tri_insertion(l):
    for i in range(1,len(l)):
        val = l[i]
        j = i
        while j > 0 and l[j-1] > val:
            l[j] = l[j-1] # décalage
            j = j - 1
        l[j] = val
        print((i,j), l)

#============================================================================
# Tri rapide
def partition(l, g, d):
    pivot = l[g]
    m = g+1
    for i in range(g+1, d):
        if l[i] < pivot:
            if m < i: # ne pas faire l'échange inutile si m = i
                l[m], l[i] = l[i], l[m]
            m = m + 1
    if m > g + 1: # ne pas faire l'échange inutile si m = g + 1
        l[g], l[m-1] = l[m-1], l[g]
    print((g,d), l, (pivot,m-1))
    return m - 1

def tri_rapide(l, g = 0, d = None):
    if d == None:
        d = len(l)
    m = partition(l, g, d)
    if g < m - 1:
        tri_rapide(l, g, m)
    if d > m + 2:
        tri_rapide(l, m+1, d)
```

```
#============================================================================
# Tri fusion
def interclassement(l, g, m, d):
    i1 = g
    i2 = m
    tmp = []
    for i in range(d - g):
        if ( i2 == d or l[i1] < l[i2] ) and i1 < m:
            tmp.append(l[i1])
            i1 = i1 + 1
        else:
            tmp.append(l[i2])
            i2 = i2 + 1
    # Recopie de tmp dans l
    for i in range(d - g):
        l[g+i] = tmp[i]

def tri_fusion(l, g = 0, d = None):
    if d == None:
        d = len(l)
    m = (g+d) // 2
    print('Appels : ', g, m, d)
    if g < m - 1:
        tri_fusion(l, g, m)
    if m+1 < d:
        tri_fusion(l, m, d)
    interclassement(l, g, m, d)
    print((g,m,d), l)

#============================================================================
# Utilisation des tris

liste_originale = [8, 2, 9, 3, 1, 5]
print('Liste originale : ', liste_originale)
tris = ['Tri sélection', 'Tri insertion', 'Tri rapide', 'Tri fusion']

for i in range(4):
    liste = liste_originale[:]
    print()
    print('***', tris[i], '***')
    if i == 0:
        tri_selection(liste)
    elif i == 1:
        tri_insertion(liste)
    elif i == 2:
        tri_rapide(liste)
    else:
        tri_fusion(liste)
    print('Liste triée :', liste)
```

Résultat

```
Liste originale :  [8, 2, 9, 3, 1, 5]

*** Tri sélection ***
(4, 0) [1, 2, 9, 3, 8, 5]
(1, 1) [1, 2, 9, 3, 8, 5]
(3, 2) [1, 2, 3, 9, 8, 5]
(5, 3) [1, 2, 3, 5, 8, 9]
(4, 4) [1, 2, 3, 5, 8, 9]
(5, 5) [1, 2, 3, 5, 8, 9]
Liste triée : [1, 2, 3, 5, 8, 9]

*** Tri insertion ***
(1, 0) [2, 8, 9, 3, 1, 5]
(2, 2) [2, 8, 9, 3, 1, 5]
(3, 1) [2, 3, 8, 9, 1, 5]
(4, 0) [1, 2, 3, 8, 9, 5]
(5, 3) [1, 2, 3, 5, 8, 9]
Liste triée : [1, 2, 3, 5, 8, 9]

*** Tri rapide ***
(0, 6) [5, 2, 3, 1, 8, 9] (8, 4)
(0, 4) [1, 2, 3, 5, 8, 9] (5, 3)
(0, 3) [1, 2, 3, 5, 8, 9] (1, 0)
(1, 3) [1, 2, 3, 5, 8, 9] (2, 1)
Liste triée : [1, 2, 3, 5, 8, 9]

*** Tri fusion ***
Appels :  0 3 6
Appels :  0 1 3
Appels :  1 2 3
(1, 2, 3) [8, 2, 9, 3, 1, 5]
(0, 1, 3) [2, 8, 9, 3, 1, 5]
Appels :  3 4 6
Appels :  4 5 6
(4, 5, 6) [2, 8, 9, 3, 1, 5]
(3, 4, 6) [2, 8, 9, 1, 3, 5]
(0, 3, 6) [1, 2, 3, 5, 8, 9]
Liste triée : [1, 2, 3, 5, 8, 9]
```