Algorithmes de tri

1 Introduction

2 Tris itératifs quadratiques

2.1 Tri à bulles

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

2.2 Tri par selection

```python
def rang_mini_from_j(L, j):
    i, m = j, L[j]
    for k in range(j+1, len(L)):
        if L[k] < m:
            i, m = k, L[k]
    return(i)

def Tri_selection(L):
    for j in range(len(L)-1):
        i = rang_mini_from_j(L, j)
        if i != j:
            L[i], L[j] = L[j], L[i]
```

2.3 Tri par insertion

```python
def insere(L, j):
    k, a = j, L[j]
    while k > 0 and a < L[k-1]:
        L[k] = L[k-1]
        k -= 1
    L[k] = a

def TriInsertion(L):
    for j in range(1, len(L)):
        insere(L, j)
```

3 Tris récursifs efficients

3.1 Tri par partition-fusion

```python
def fusion(L1, L2):
    """fusionne deux listes triées"""
    if L1 == []:
        return L2
    if L2 == []:
        return L1
    #Désormais, les deux listes sont non vides.
    if L1[0] <= L2[0]:
        return([L1[0]] + fusion(L1[1:], L2))
    else:
        return([L2[0]] + fusion(L1, L2[1:]))


def TriFusion(L):
    if len(L) <= 1:
        return L
    m = len(L)//2
    return(fusion(TriFusion(L[:m]), TriFusion(L[m:])))
```

3.2 Tri rapide *quick sort*

```python
def quicksort(L):
    if len(L) <= 1:
        return L
    else:
        pivot = L[0]
        plus_grand = []
        plus_petit = []
        for x in L[1:]:
            if x >= pivot:
                plus_grand.append(x)
            else:
                plus_petit.append(x)
        return(quicksort(plus_petit) + [pivot] + quicksort(plus_grand))
```

4 Autres tris

4.1 Tri par comptage

```python
def comptage (L,n) :
    P = [0 for i in range(n)]
    for k in L :
        P[k] += 1
    return(P)


def TriComptage (L,N) :
    M = []
    P = comptage(L,N)
    for k in range(N) :
        for i in range(P[k]) :
            M.append(k)
    return(M)
```