Tri par insertion

```
Procédure TriInsertion(T : tableau[0..n-1] d'entiers)
/* Trie en ordre croissant le tableau passé en paramètre */
lexique i, j, x : entiers
{
    pour i allant de 1 à n—1 faire {
        x := T[i]
        j := i
        tant que j > 0 et T[j-1] > x faire {
            T[j] := T[j-1]
            j := j—1
        }
        T[j] := x
    }
}
```

Tri rapide (1)

```
Procédure tri_rapide_aux(T : tableau [0..n-1] d'entiers ;
                         premier, dernier : entiers)
lexique pivot : entier
{
    si premier < dernier alors {
        pivot := choix_pivot(T, premier, dernier)
        pivot := partitionner(T, premier, dernier, pivot)
        tri_rapide_aux(T, premier, pivot-1)
        tri_rapide_aux(T, pivot+1, dernier)
    }
}
```

Tri rapide (2)

```
Fonction partitionner(T : tableau [0..n-1] d'entiers ;
                      premier, dernier, pivot : entiers)
lexique i, j : entiers {
    échanger(T[pivot], T[dernier])
    j := premier
    pour i de premier à dernier - 1 faire
        si T[i] ≤ T[dernier] alors {
          échanger(T[i], T[j])
          j := j + 1
        }
    échanger(T[dernier], T[j])
    pivot := j
    retourner pivot
}
```

Tri fusion (1)

```
Procédure tri_fusion_aux(T : tableau [0..n-1] d'entiers ;
                         g, d : entiers)
lexique m : entier {
    si d > g alors {
        m := (g + d) / 2
        tri_fusion_aux(T, g, m)
        tri_fusion_aux(T, m + 1, d)
        fusionner(T, g, m, d)
    }
}
```

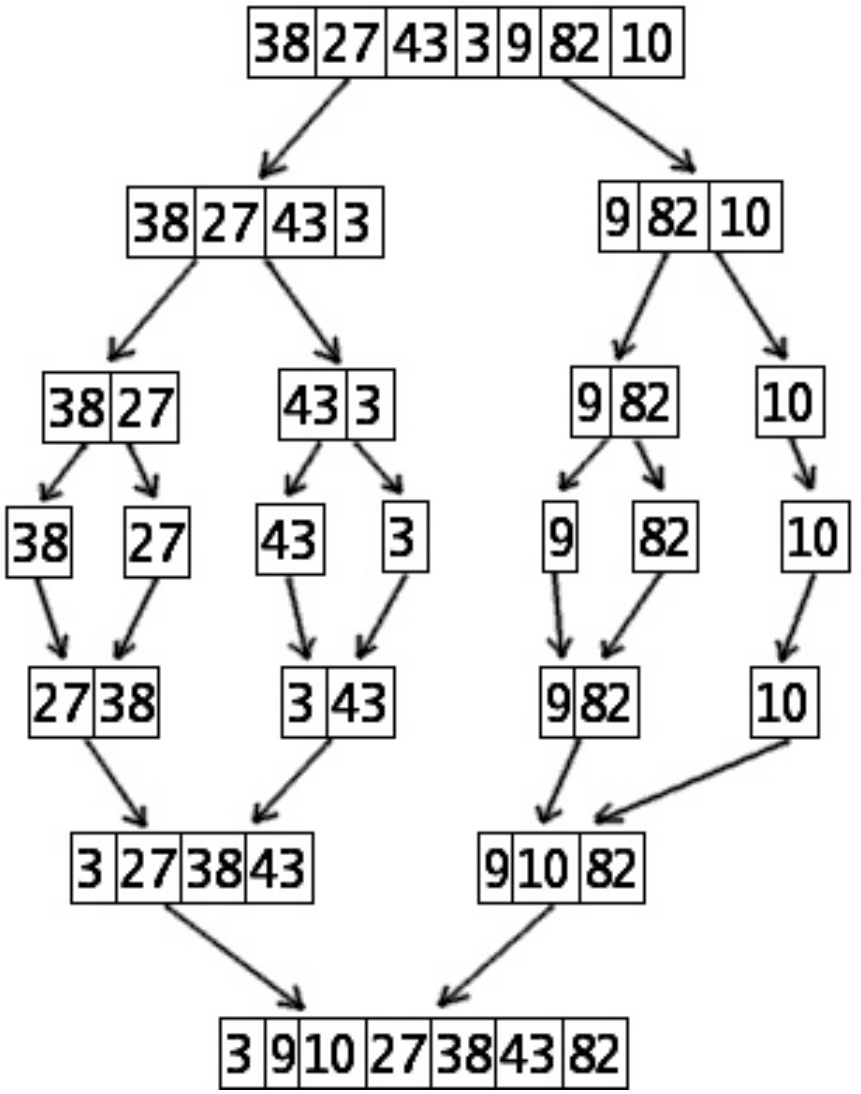
Tri fusion (2)

```
Procédure fusionner(T : tableau [0..n-1] d'entiers ;
                    gauche, milieu, droite: entiers)
lexique i, j, k : entiers ;
        U : tableau [0..n-1] d'entiers
{
    i := 0
    j := gauche
    k := milieu + 1

    tant que i ≤ droite – gauche faire {
        si j ≤ milieu et k ≤ droite alors {
            si T[j] ≤ T[k] alors { U[i] := T[j] ; j := j+1 }
            sinon                { U[i] := T[k] ; k := k+1 }
        } sinon {
            si       j > milieu alors { U[i] := T[k] ; k := k+1 }
            sinon si k > droite alors { U[i] := T[j] ; j := j+1 }
        }
        i := i+1
    }

    pour i allant de 0 à droite – gauche faire
        T[gauche+i] := U[i]
}
```

Tri fusion


38 27 43 3 9 82 10
38 27 43 3
9 82 10
38 27
43 3
9 82
10
38
27
43
3
9
82
10
27 38
3 43
9 82
10
3 27 38 43
9 10 82
3 9 10 27 38 43 82