Tri fusion

```
def TriFusion(T) :
    Si |T| > 1 Alors :
        Si |T| == 2 Alors :
            Si T[0]>T[1] Alors T[0]  ↔ T[1]
        Sinon :
            m = |T| / 2
            T = Fusion(TriFusion(T[0..m-1]),TriFusion(T[m..|T|-1]))
    Retourner T
```

Fusion = fusion des deux tableaux donnés en argument.

Tri fusion

```
def Fusion (L1,L2) :
    res = [ ]
    i1 = i2 = 0
    Tant que i1< |L1|  ∧  i2 <  |L2| :
        Si L1[i1] < L2[i2] Alors :
            res.append(L1[i1])          → on ajoute L1[i1] à res…
            i1 = i1+1
        Sinon :
            res.append(L2[i2])          → on ajoute L2[i2] à res…
            i2 = i2+1
    Si i1== |L1| Alors :
        res += L2[i2:]                  → on concatène la fin de L2 à res…
    Sinon :
        res += L1[i1:]                  → on concatène la fin de L1 à res…
Retourner res
```

“ET” logique ⚠️

Tri fusion

Java

```
public static void triFusion (int [] T, int d, int f)
{
    int m;
    if(d<f)
// si d>f, T est vide, si d==f, T ne contient qu'un element
    {
        m = (d+f)/2;
        triFusion(T, d, m);
        triFusion(T, m+1, f);
        fusionner (T, d, m, f);
    }
}
```

Tri fusion

Java

```
// les zones a fusionner sont: d,d+1,...,m   et m+1,m+2,...,f
public static void fusionner (int T[], int d, int m, int f)  {
    int [] AuxZone = new int[f-d+1];
    for (int j=0;j<AuxZone.length;j++) // on recopie la zone dans AuxZone...
        AuxZone[j]=T[d+j];
    int i1 = 0,   i2 = (m-d)+1, i=d ;
    while ((i1<= (m-d)) && (i2 <= f-d)){
          if(AuxZone[i1] <= AuxZone[i2]){
               T[i] = AuxZone[i1];
               i1++;         }
          else {
               T[i] = AuxZone[i2];
               i2++;}
          i++;}
    while(i1<=(m-d)) { // on met la fin de la premiere moitie si il en
reste...
        T[i]=AuxZone[i1];  i1++;  i++;  }
    while(i2<=f-d)  { // idem pour l'autre moitie…
        T[i]=AuxZone[i2];  i2++;  i++;  }
}
```