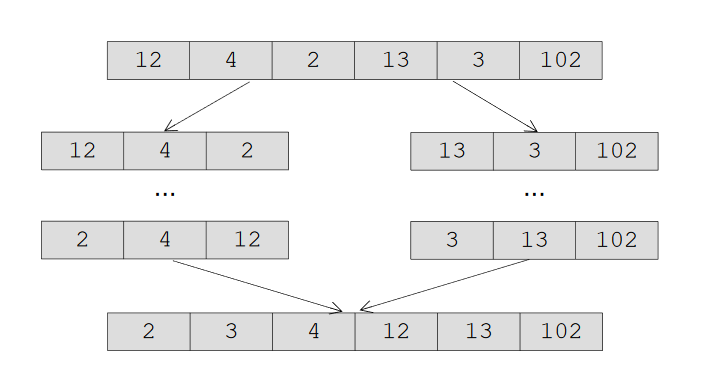
**Documentation sur le tri fusion et le tri rapide**

**Tri fusion :**

on coupe le tableau en deux et on trie chacune des sous-parties (diviser pour régner), puis on reconstitue le tableau entier en fusionnant les deux parties et en respectant l’ordre.



**Code en algo :**

// fusion des parties de tab d'indices dans [a,b] et [b+1,c] avec a<b<=c

// les elements de ces parties sont tries en ordre croissant

fonction sans retour fusion(entier tab[],entier a, entier b, entier c)

entier i, j, k, t[c-a+1];

**début**

**pour** (i allant de 0 à t.longueur-1 pas de 1 **faire**

t[i] <- tab[a+i];

**finpour**

i <- 0, j <- b-a+1, k <- a;

**tantque**(k <= c){si (i <= b-a et (j = c-a+1 ou t[i] <= t[j])) **alors**

tab[k] <- t[i];i++;

**sinon**

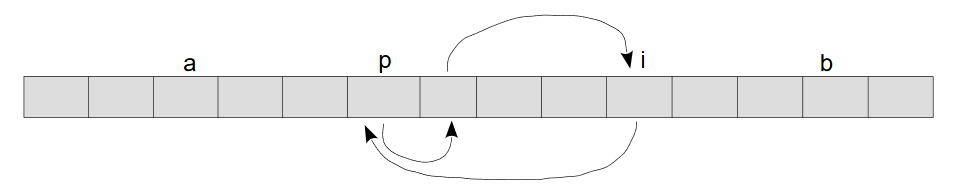
tab[k] <- t[j];j++;finsik++;

**fintantque**

**fin**

**Tri rapide :**

on trie un élément (dit pivot) en déplaçant à sa gauche les éléments plus petits et à sa droite les éléments plus grands. Puis on recommence l’opération sur les deux sous parties gauche et droite.



**Code en algo :**

fonction sans retour triRapide(entier tab[])

entier pile[tab.longueur];

entier top = 1, a, b, p

**début**

pile[0] <- 0;

pile[1] <- tab.longueur-1;

**tantque** (top>=0) **faire**

a <- pile[top-1];

b <- pile[top];

top <- top-2;

p <- partition(tab,a,b);

**si** (p-1>a) **alors**

pile[top+1] <- a;

pile[top+2] <- p-1;

top <- top+2;

**finsi**

**si** (p+1<b) **alors**

pile[top+1] <- p+1;

pile[top+2] <- b;

top <- top+2;

**finsi**

**fintantque**

**Fin**