```
// Usage: sort(a,i,j,b);
// Pre:   0 <= i <= j <= a.length.
//        a != b, a.length==b.length.
// Post:  The section a[i..j) has been sorted in ascending
//        order. The rest of a is unchanged.
//        The same section in b has been modified.
public static void sort( int[] a, int i, int j, int[] b )
{
    if( j-i < 2 ) return;
    int m = i+(j-i)/2;
    sort(a,i,m,b);
    sort(a,m,j,b);
    int p = i, q = m, r = i;
    while( p != m && q != j )
        // i <= p <= m <= q <= j.
        // a[i..m) and a[m..j) are in ascending order and
        // contain otherwise the same values as before the
        // call.
        // b[i..r) is in ascending order and contains the
        // same values as a[i..p) and a[m..q).
        // All values in b[i..r) are <= all values in both
        // a[p..m) and a[q..j).
    {
        if( a[p] <= a[q] ) b[r++] = a[p++];
        else               b[r++] = a[q++];
    }
    System.arraycopy(a,p,b,r,m-p);
    System.arraycopy(a,q,b,r,j-q);
    System.arraycopy(b,i,a,i,j-i);
}
```

Sort first and second half

Merge the two sections into b

Only one of those two calls will move data because either `m-p==0` or `j-q==0`

Copy back into a

```
public static void sort( int[] a, int i, int j, int[] b )
{
    if( j-i < 2 ) return;
    int m = i+(j-i)/2;
    sort(a,i,m,b);
    sort(a,m,j,b);
    int p = i, q = m, r = i;
    while( p != m && q != j )
    {
        if( a[p] <= a[q] ) b[r++] = a[p++];
        else               b[r++] = a[q++];
    }
    System.arraycopy(a,p,b,r,m-p);
    System.arraycopy(a,q,b,r,j-q);
    System.arraycopy(b,i,a,i,j-i);
}
```