# Sorted Merge (10.1)

Constraints:

* What sizes do the arrays have? Arbitrary sizes
* Do we know how many elements are in A?

Idea:

* Move all elements of array A to the end of this array
* As long as there are untested elements of either array:
  + Compare the smallest element of array A with the smallest element of array B, and put the smaller one in the next free position in array A
* When one of the array has no more elements:
  + Move the remainder of the other array to the end of array A (if it is not already there, in the case that B ran out of elements first)

Complexity:

* Move all elements of A to end of array: A
* Move all the elements of A and B once: A+B
* Thus, there are 2A+B moves in the worst case
* Complexity: O(A+B)

Implementation:

class SortedMerge {

public static void main(String[] args) {

int[] b = {3, 5, 7, 10};

int[] aTmp = { 4, 6, 7, 10, 11 };

int[] a = new int[aTmp.length + b.length];

for (int i = 0; i < aTmp.length; i++)

a[i] = aTmp[i];

merge(a, b);

System.out.println(a);

}

private static void merge(int[] a, int[] b) {

int i = moveToEnd(a, a.length - b.length);

int j = 0;

int k = 0;

while (i < a.length && j < b.length) {

if (a[i] < b[j])

a[k] = a[i++];

else

a[k] = b[j++];

k++;

}

if (j < b.length)

moveToEnd(b, j, a);

}

private static int moveToEnd(int[] a, int size) {

for (int i = 0; i < size; i++) {

a[a.length-1-i] = a[size-1-i];

}

return a.length-size;

}

private static int moveToEnd(int[] from, int index, int[] to) {

int n = from.length;

for (int i = 0; i < n; i++) {

to[to.length-n+i] = from[index+i];

}

}

}