Iterative version of Merge Sort

/**
   This method assumes that the following portions of the array a are sorted:
   from a[first] to a[middle] and from a[middle + 1] to a[last].
   The method then merges these two sub-arrays, so that a is sorted
   from a[first] to a[last]
*/
public static void merge(int[] a, int first, int middle, int last)
{ int[] temp = new int[last - first + 1];
  int next = 0; // next item is to inserted in temp[next]
  int k = first;
  int m = middle + 1;

  while(true)
  { // left portion all in proper place, copy the remaining right portion
    if(k > middle)
    { System.arraycopy(a, m, temp, next, last - m + 1); break;}

    // right portion all in proper place, copy the remaining left portion
    if(m > last)
    { System.arraycopy(a, k, temp, next, middle - k + 1); break;}

    // insert the next item in the proper place
/\* Compare a[k] with a[m] and insert the smaller into temp[next], then next++ */
if (a[k] <= a[m])
    { temp[next++] = a[k]; k++;}
else
    { temp[next++] = a[m]; m++;}

count++;  // the only place count is incremented

}  // Copy the temp array back to a
System.arraycopy(temp, 0, a, first, last - first + 1);
Iterative version of mergesort. Successively merge subarrays of size \( \text{incr} \). The process starts with \( \text{incr} = 1 \). The algorithm goes through the array, merging two successive subarrays, each of length \( \text{incr} \). By induction, both parts are already sorted. There might be some irregularities at the end of the array, if \( 2 \times \text{incr} \) does not divide the size of the array. More or less obvious modifications have to be made in that case. Then, the \( \text{incr} \) is doubled, and the process continues until \( \text{incr} \) is \( \geq \) array size, when the process stops.

```java
public static void mergeSort(int[] a)
{
    int lasta = a.length - 1; // last index of a
    int incr = 1; // start with bunch of sorted subarrays, each of length 1
    while (incr <= lasta)
    {
        /* start to merge subarrays of size \( \text{incr} \) at the location \( \text{begin} \)

        That is the sorted subarrays

        \( a[\text{begin}], a[\text{begin} + 1], \ldots, a[\text{begin} + \text{incr} - 1] \)

        and

        \( a[\text{begin} + \text{incr}], \ldots, a[\text{begin} + 2 \times \text{incr} - 1] \)

        are merged

        */
        int begin = 0;
```
while (begin <= lasta)
    {  // two sub-arrays, each of size incr are to be merged
        if (begin + 2*incr - 1 <= lasta)
            {  merge(a, begin, begin + incr - 1, begin + 2*incr - 1);
                begin += 2*incr;
            }

        // two sub-arrays, one of length incr and another shorter are to merged
        else if (begin + incr - 1 < lasta)
            {  merge(a, begin, begin + incr - 1, lasta);
                begin = lasta + 1;
            }

        /* One array is left, it is of length \leq incr,
           it will be merged next iteration */
        else begin = lasta + 1;
    }
}
incr *= 2;  // double the increment