# Mergesort

**Merging** Two sorted arrays $A[1, \ldots, n_1]$ and $B[1, \ldots, n_2]$ can be merged together and written (sorted) into an array $C[1, \ldots, n_1 + n_2]$ in $O(n_1 + n_2)$ worst case time. This is done by walking through the two arrays in parallel, keeping a pointer to the smallest unmoved item in each list. At each step it

- compares the smallest remaining items in each of the two lists
- moves (writes) the smaller of the two to the next place in $C$
- advances by one the pointer that had pointed to that smallest

**Mergesort Description.** Given an array $A[1, \ldots, n]$ of $n$ numbers, mergesort first recursively sorts $A[1, \ldots, n/2]$ and $A[n/2, \ldots, n]$ and then merges the two sorted subarrays into one sorted array. Note that Mergesort requires extra memory into which to originally write the new merged list before writing it back into the original array,

**Recurrence.** Mergesort is a typical divide-and-conquer algorithm, which can usually be analyzed using a recurrence. Let $T(n)$ denote the worst-case running time of mergesort on $n$ numbers.

The two recursive calls of mergesort take $2T(n/2)$ time. The merging step takes $O(n)$ time to merge the two sublists together. The recursion does not go on forever. It bottoms out when $n = 1$ in which case there is nothing to be done because an array of one number is trivially sorted. This gives the following recurrence:

$$\begin{aligned} T(1) &= O(1) \\ T(n) &= 2T(n/2) + O(n) \end{aligned}$$

which we have already seen implies $T(n) = O(n \log n)$.

*Note: This document was written by M J. Golin, revised from an original by S.W. Cheng, for COMP3711H, HKUST.*