7. Sorting Algorithms :: Merge Sort :: Example

Merge sort uses a beautifully simple idea and is typically implemented using recursion, although we can implement it using repetition (a loop) as well.

Before looking at the pseudocode, let's discuss what we mean by **merge**. Suppose that we have two sorted (in ascending order) lists a and b:

$a = \{2, 3, 5, 7, 11, 13\}$ $b = \{2, 4, 6, 8\}$

and we wish to combine these two lists to create a new list c where the elements in c will be in ascending order:

$c = \{2, 2, 3, 4, 5, 6, 7, 8, 11, 13\}$

Here is the procedure:

$a = \{2, 3, 5, 7, 11, 13\}$ $b = \{2, 4, 6, 8\}$

c = { }

To summarize, here's what we did:

1. Compared the first non-crossed-out elements of a and b.
2. Copied the smaller element to the end of c. When the elements are equal, it did not matter which one we copy to c, so we chose to copy from a.
3. Crossed out the element in a or b.
4. Repeated Steps 1–3 until all of the elements in b were crossed-out.
5. Copied the remaining elements of a to c.

7. Sorting Algorithms :: Merge Sort Example (continued)

This procedure is known as a **merge** and it is what gives merge sort its name. But how did the elements of a and b get sorted—we kinda glossed over that. The list merging process is only part of the merge sort algorithm. To see how the complete algorithm works, let's start with this list (to be sorted into ascending order) which has $n = 10$ elements.

$$list = \{4, 2, 7, 3, 5, 13, 11, 8, 6, 2\}$$