8. Sorting Algorithms :: Merge Sort

Before looking at the pseudocode, here is a list of the rules the algorithm must implement:

1. If a (parent) list *list* has 2 or more elements then split it into two child lists: $list_{\mathrm{L}}$ containing the first $\lfloor n/2 \rfloor$ elements of *list* and $list_{\mathrm{R}}$ containing the last $\lceil n/2 \rceil$ elements.
2. If a list has 1 element, it is trivially sorted so we do nothing,
3. When two child lists become sorted, merge them to form a sorted parent list.
4. Repeat these rules until the original list is sorted.

8. Sorting Algorithms :: Merge Sort and Recursion

Merge sort begs to be solved by recursion. Remember from *Recursion* : *Section 3* that we discussed seven characteristics that must be present for a problem to be solvable by recursion:

1. The size of the problem must be reducible to a smaller, basically-equivalent subproblem.

   This is what we do when we split a list of size n *into two child sublists each roughly of size* n/2*.*

2. The smaller, basically-equivalent subproblem must be simpler to solve than the larger problem.

   It is faster (easier) to sort a smaller list than a larger list.

3. The solution to the original problem requires repetition.

   Rule 5 specifies that we repeatedly perform these rules until the original list is sorted.

4. There must be a base case at which point the size of the problem cannot be further reduced.

   The base case is when the size of a list is 1 which cannot be reduced to a smaller list.

5. The solution to the base case is generally easily-obtainable.

   A list of size 1 is trivially sorted.

6. The solution to the smaller subproblem must be returned and used in solving the larger problem.

   The sorted child lists are merged to form larger sorted lists.

7. The solution to the original problem results from solving all of the subproblems.

   Eventually, we end up with only two sorted lists which are merged to form the sorted original list.