17. Sorting Algorithms :: Quick Sort :: Partitioning the Partitions

Once we have partitioned *list* into two sublists (by choosing the first element as the pivot):

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

what would happen if we partition each of these sublists in the same way?

$list_{\mathrm{L}} = \{2, 2, 3\}$ $list_{\mathrm{R}} = \{7, 5, 13, 11, 8, 6, 4\}$

17. Sorting Algorithms :: Quick Sort

Since partitioning does all of the work of actually moving the elements around to form the sorted list, the main quick sort method is very simple: we simply partition the input list and recursively quick sort the left and right sublists:

```
Method quickSort(In: List<T> list, In: fromIndex, In: toIndex)
  -- The base case is reached when the list has only one element, i.e., when
  -- fromIndex is greater than or equal to toIndex. In this case, we have a list
  -- that is trivially sorted, so we have nothing to do.
  If fromIndex ≥ toIndex Then Return
  -- Otherwise, partition the list into a left half and a right half such that
  -- all of the elements in the left half are less than the elements in the right
  -- half.
  partitionIndex ← partition(list, fromIndex, toIndex)
  -- Recursively quick sort the left and right halves.
  quickSort(list, fromIndex, partitionIndex)
  quickSort(list, partitionIndex + 1, toIndex)
End Method quickSort
```

To quick sort an ArrayList we would call *quickSort*() this way:

```
ArrayList<Integer> list = new ArrayList<>();
// Integers are added to list...
quickSort(list, 0, list.size() - 1);
```