I am using JDK-8 (x64). For Arrays.sort I found the following in the Java documentation:

The sorting algorithm is a Dual-Pivot **Quicksort** by Vladimir Yaroslavskiy, Jon Bentley, and Joshua Bloch.`

For Collections.sort I found this:

This implementation is a stable, adaptive, iterative **mergesort** ... This implementation **dumps the specified list into an array, sorts the array**, and iterates over the list resetting each element from the corresponding position in the array.

If Collections.sort uses an array, why doesn't it just call Arrays.sort or use dual-pivot **QuickSort**? Why use **Mergesort**?

The API guarantees a stable sorting which Quicksort doesn’t offer. However, when sorting primitive values by their natural order you won’t notice a difference as primitive values have no identity. Therefore, Quicksort is used for primitive arrays as it is slightly more efficient.

For objects you may notice, when objects which are deemed equal according to their equals implementation or the provided Comparator change their order. Therefore, Quicksort is not an option. So a variant of MergeSort is used, the current Java versions use TimSort. This applies to both, Arrays.sort and Collections.sort, though with Java 8, the List itself may override the sort algorithms.

Quick Sort has two major drawbacks when it comes to merge sort:

* It's not stable while it comes to non primitive.
* It doesn't guarantee n log n performance.