6. Sorting Algorithms :: Insertion and Selection Sort in Practice

We have seen that insertion sort and selection sort are both $O(n^2)$ algorithms, but neither is particularly efficient for large lists. For example, suppose each comparison operation in selection sort requires 25 nanoseconds (a nanosecond is one-one billionth of a second, so 25 nanoseconds is 25×10^{-9} seconds). Let $t(n)$ be the time to sort a list of size n, where $c(n)$ is the number of comparisons relative to n:

n	$c(n)$	$t(n)$ sec	$t(n)$ min	$t(n)$ hr	$t(n)$ day	$t(n)$ years	$t(n)$ millennia
10^2	10^4	2.5×10^{-4}					
10^3	10^6	2.5×10^{-2}					
10^4	10^8	2.5×10^{0}					
10^5	10^{10}	2.5×10^{2}	4.2×10^{0}				
10^6	10^{12}	2.5×10^{4}	4.2×10^{2}	6.9×10^{0}			
10^7	10^{14}	2.5×10^{6}	4.2×10^{4}	6.9×10^{2}	2.9×10^{1}		
10^8	10^{16}	2.5×10^{8}	4.2×10^{6}	6.9×10^{4}	2.9×10^{3}	7.9×10^{0}	
10^9	10^{18}	2.5×10^{10}	4.2×10^{8}	6.9×10^{6}	2.9×10^{5}	7.9×10^{2}	
10^{10}	10^{20}	2.5×10^{12}	4.2×10^{10}	6.9×10^{8}	2.9×10^{7}	7.9×10^{4}	7.9×10^{1}

Now, its not every day we sort an ArrayList<Integer> list of 10-billion elements, but clearly if we ever need to, we had better be prepared to wait a very, very long time for the selection sort algorithm to finish. However, consider sorting a reasonably-sized list of 1-million elements: even that is going to take almost 7 hours, and 4 minutes for a small list of 100,000 elements is completely unreasonable. Therefore, over the last 70-some years, much effort has been expended to find the best sorting algorithms and in the next few sections we will look at two of the best: merge sort and quick sort.