Complexity Order

In this program, 3 separate sorting algorithms are being used simultaneously. To derive the complexity order of the entire program, it is necessary to analyze each individual sorting algorithm.

The first one is insertion sort. Because in the worst-case scenario, where an array is given in descending order, will have N*N operations, we say that insertion sort has a complexity of O(N^2).

The second one is quick sort. Although, on average, quick sort behaves like an N log(N) algorithm, it does have an N*N worst-case scenario, where the pivot that gets chosen is the worst possible one, where only a single element gets sorted per pass.

Lastly, we have the merge sort algorithm, which best case, worst case, and average cases are all the same, N log(N). This makes merge sort the most predictable one of the three.

When all three algorithms are combined, their worst-case scenario becomes O(N^2). However, since the program is a race that ends as soon as 1 of the 3 finishes, the complexity needs only to match the best performing of the 3's worst-case scenario, which is merge sort's N log(N). This makes the complexity of the program execute, in its worst case, 3N log(N), but since 3 is a constant, the analysis drops it, and the program's complexity order becomes O(N log(N)).