For each of the files one through three, numbers of 50, 500, and 5,000 lines were tested with both the regular QuickSort algorithm as well as a modified version of QuickSort which delegated sublists of less than 50 items to be sorted with InsertionSort instead. For file1, the threshold at which QuickSort becomes less efficient than its modified counterpart is immediate. It will always be slower than the other program. For file2, the same applies as well. For file3, however, the user can receive the same results from either file up until more than 500 lines must be processed.

Each of the files represents different cases for the algorithms to sort though. In file1, the numbers are randomly sorted and need longer in order to be sorted into smaller lists and then reassimilated. In file2,the numbers are in reversed order and in file3, the numbers are already sorted. This being said, the numbers in file3 only have to be scanned once and not sorted. In any of these cases, the smaller sublists (less than 50 numbers) are sorted faster with InsertionSort, except file3, where the only required operation has the same runtime in either InsertionSort or QuickSort for up to 500 items. Therefore, I'd say that the modified QuickSort will always be more efficient than the original, considering the only other counter example was that of the best case scenario.