The choice of sorting and searching algorithms are very important for the performance of the program.

- Insertion sort has a time complexity of O(n) in the best case scenario, because the data is already sorted and only needs to be compared once per value. However, on average the complexity is O(n^2) because a nested loop is used to compare each value multiple times.

- Merge sort always has a time complexity of O(n log n) because it is a divide and conquer technique similar to binary search.

- Linear search has a complexity of O(1), if the value being searched is the very first value only 1 comparison must be made. However, it typically has a time complexity of O(n), because the number of comparisons is usually within the magnitude of the number of terms.

- Binary search is a divide and conquer method so it has a time complexity of O(log n) on average. In the best case scenario it can be O(1) if the first value checked is the target value.

In terms of searching algorithms, the larger the data set becomes, the faster binary search is compared to linear search. Linear search typically has a time complexity of O(n), and as the values become very large this could take a very long time. However, binary search has a complexity of O(log n), which will be much quicker than linear search when n is a large value. For small data sets, linear search will likely be faster, but as the dataset grows binary search will be far more efficient. A better search method increases efficiency simply because less computations will be done, and this will allow the program to finish quicker.