Searching and sorting is probably one of the most important things you can learn to do in programming. Deitel and Deitel (2019) describe searching data as determining whether a value (referred to as the search key) is present in the data and determining its location if it is. Sorting is described as placing data in ascending or descending order, based on one or several sort keys (Deitel & Deitel, 2019). Sorting is important in programming for the same reasons it is important in real life: it is monumentally faster and easier to locate data in a sorted list than an unsorted list (Kaushal, 2017). Similar to sorting, searching is just as important in computer science because almost always a user or programmer will be asking for some data, and the computer then has to search for it, an issue that is exemplified by the massive amounts of data a computer can store (Megharaja, Rakshitha, & Swetha, 2018).

Sorting and searching may be necessary in designing software applications such as in account creation for an app: users creating an account would have to use a username that has not been used before. In order for this to work efficiently, your program would need to be able to sort the list of usernames that already exist alphabetically, and that way it makes it easier to search and see if said username has already been used. Another example of when sorting and searching is necessary to use in software applications could be seen in profanity filters. Profanity filters are necessary to assert that users are complying with guidelines when using applications or other services. As such, being able to sort and search through banned words and phrases quickly is essential to providing a safe and respectful environment to users.

According to Deitel and Deitel (2019), selection sort is a simple but inefficient sorting algorithm where the first iteration selects the smallest element in the array and swaps it with the first element, then does the second smallest element and swaps it with the second element of the array, and so on and so forth. Insertion sort is another simple but inefficient sort where each element in the array is checked against the values that precede it and puts it in the correct location (Deitel & Deitel, 2019). Merge sort is a far more efficient sorting algorithm than the previous two. In merge sorts, an array is split into two equally sized arrays that are sorted and then are merged into one larger array (Deitel & Deitel, 2019).

Linear search is probably one of the more common searches you see that is representative of how you may search for something in real life. Linear search involves an algorithm searching through the elements of an array and comparing that to the value you are trying to find (Deitel & Deitel, 2019). Binary search is a much more efficient search, where first the middle element is compared against the searched value. If it is the right value, then the search ends, otherwise, the array is split into two subarrays, where the middle element of the correct sorted array is once against compared to the searched value, going on until the correct array element is found (Deitel & Deitel, 2019)

As for the best sorts from the 3 discussed, merge sort is easily the best choice. Selection and insertion sorts are very inefficient sorts since they often contain two loops in them to run correctly (Deitel & Deitel, 2019). This results in both sorts having worst case scenarios of taking n^2 time to run, which exceptionally poor (Deitel & Deitel, 2019). Merge sort, on the other hand, takes only about n log n time to run, which is a much faster time than n^2, even though it is conceptually more complex (Deitel & Deitel, 2019).

As for the better search, binary search easily takes the win. Linear search has a worst case scenario of taking n time to run, as the value you are searching for may be the last value in the array (Deitel & Deitel, 2019). Binary search, however, takes log n time to run, which is much faster, even though it requires the array to be sorted beforehand.

<http://www.ijsrd.com/articles/IJSRDV5I70074.pdf>

Megharaja D.S, Rakshitha H J, & Shwetha K. (2018). Significance of Searching and Sorting in Data Structures. *International Research Journal of Engineering and Technology (IRJET).* 5, 294-297. <https://www.irjet.net/archives/V5/i4/IRJET-V5I459.pdf>

Kaushal R. (2017). Why Sorting is So Important in Data Structures. *International Journal for Scientific Research & Development*. 5(7), 114-117. <http://www.ijsrd.com/articles/IJSRDV5I70074.pdf>

Deitel, P., & Deitel, H. (2019). *Intro to Python for Computer Science and Data Science: Learning to Program with AI, Big Data and The Cloud* (1st ed.). Pearson.