Algorithms Chapter 8 Assignment – Kelling

Like Merge Sort, Quick Sort also uses the divide and conquer approach. However, in Merge Sort most of the work isn’t being done in the divide step, but is done during the combine step. Quick Sort on the other hand performs all of the real work in the divide step, with no real functionality occurring in the combine step. (Technically, Quick Sort doesn’t really have a ‘combine’ step)

**Quick Sort’s Cases** 🡪 best: O(n log(n)), average: O(n log(n)), worst: O(n^2).

**Merge Sort’s Cases** 🡪 best: O(n log(n)), average: O(n log(n)), worst: O(n log(n)).

Even though Quick Sort has O(n^2) in worst case, this can usually be avoided depending on your pivot selection. It’s cache performance is much higher than other sorting algorithms, and the operations of its inner loop are much simpler. Therefore, when implemented appropriately Quick Sort is generally a much faster sorting algorithm, and especially on an unsorted array. Merge Sort performs irrespectively the same whether the data is sorted or not. Quick Sort also performs better when the data is stored in memory, whereas when the data is stored on an external device, Merge Sort would be the clear winner in terms of speed. Despite its speed, I think overall Quick Sort is much more complicated than Merge Sort, and for this reason someone aiming for simplicity and maintainability might choose Merge Sort instead.

**Quick Sort**:

Divide:

* Choose any element in array, call this element pivot
* Rearrange array elements so that:
* all values less than the pivot are on its right
* all values greater than the pivot are on its left
* Now 🡪 still have an unsorted list, but with elements that are generally on the correct side of the pivot

Conquer:

* Recursively sort the two subarrays:
* elements to the left of the pivot (less than)
* elements to the right of the pivot (greater than)

The Bubble Sort Algorithm is often known as the ‘algorithm not to use’. To my knowledge, it doesn't get used very much in the real world. It has bad (O(n^2)) worst case and average performance. From my understanding it has a decant best case performance, but this is only when you know the data is almost sorted. I think that there are plenty of other better suited algorithms that have this property, with better worst and average case performances. Even if Bubble Sort isn’t a very efficient and practical algorithm to use in real life situations, I think it's a great learning tool to utilize. It’s often the first, or one of the first, algorithms to be introduced in a class setting. This is because it is very easy to understand, and it’s also quite fast to implement and see the results.