Chapter 8 – Sorting

Samantha Lowell

Quicksort, like a merge sort, starts by splitting an array in half. The quicksort differs by using a pivot point, which is the node in the middle of the array. Everything less than the pivot point goes to the left array and everything greater than the pivot point goes in the right array. This is repeated using recursion, continually splitting the arrays, until everything is in order. The speed at which a quicksort works depends on where the pivot value falls in the list. The worst case is if the pivot value is either the highest or lowest value. This causes the split to be uneven and can end up taking O(n^2) time. Quicksort works best when the pivot point divides list into nearly equal pieces each time. This works best because parts of the list do not end up being compared more than once like in the worst-case scenario. As a result the algorithm takes O(n log n) time.

The worst-case scenario for a merge sort is when all of the values end up having to be moved. This results in the algorithm taking O(n log n) time, like the quicksort. If the list just happens to be mostly sorted already unsurprisingly the algorithm can perform more quickly.

A potential drawback of using quicksort is that it is not stable. This means that if there are two nodes of equal value they may not maintain the same order when sorted. Merge sort is stable and so may be a better choice if that is a concern. Merge sort works better with linked lists. When sorting arrays a merge sort requires more memory space than quicksort.

I found this animation online that compares different sorting algorithms (<https://www.toptal.com/developers/sorting-algorithms>). Just from watching the animations bubble sort seems useful when the list is nearly sorted. I am not sure how often that comes up in real life but I am sure it does. It seems that the general consensus online is that there isn’t much use for a bubble sort. People seem to prefer an insertion sort for nearly sorted or very small lists which seems to be the only times an O(n^2) sorting algorithm is useful.