Tor Parawell
Rene German
CPSC-350 Data Structures
December 19, 2020

Thoughts On Sorting Algorithms

When analyzing the times of the five different sorting algorithms, I didn't notice that big of a difference in runtime. They were minute, but when you're dealing with millions or maybe billions of data points, saving fractions of seconds at a time becomes worth it. The short comings of this analysis is that we haven't tested very many different cases. We have only used these algorithms in one way. I've included my thoughts on each individual algorithm below.

Insertion Sort – Because there are two loops, for each outer loop execution. The inner loop needs to analyze all the inner elements in the sorted part. As there becomes more elements in the array, it will take longer to sort, especially if there are a lot of unique values.

Quick Sort – It's a divide and conquer algorithm. It did not perform great when sorting a large amount of data. It has advantages because it doesn't require any extra storage, but it seemed to take on more than it could handle with too many data points.

Selection Sort – Sorts an array by repeatedly finding the minimum element from unsorted part and put it to the beginning. For every iteration, the minimum element from the unsorted subarray is picked and moved to the sorted subarray. This operation is useful for when you're on memory constraints, since unlike some other algorithms, it needs no extra space. This is also easy to implement and very easy to understand. In speed, selection sort came in about the middle of the pack, but I would imagine with huge data sets, it would be pretty slow.

Merge Sort – Pretty good for sorting a large amount of data. This recursive sorting algorithm is more memory consuming since you have to create and allocate extra space, increasing the runtime of the algorithm. In my program, this was the fastest for large data sets, but slowest in small data sets.

Bubble Sort – A simple algorithm to implement, it's more optimized the closer the array is to being sorted since it just swaps elements. It's time was pretty much always middle of the pack, only really great if the array is close to being sorted, otherwise it's average.