Willem Kroeger

CS 300: Data Structures and Algorithms: Analysis and Design

Southern New Hampshire University

January 15, 2023

Creating the code for the Selection sort algorithm was simple. I built two loops that iterate over a data set, and compare each other at each point, with a “lowest seen value” saved to the side. At the end of the iteration, if a lower value was seen by the second loop, it swaps the value in the first loop with the lower value and iterates the first loop. Simple, but inefficient because it goes through the entire data set once through the 2nd loop before iterating the 1st loop. Its speed would be the square of the number of data points it has to sort.

The quicksort algorithm took me a while to understand, but it was orders of magnitude faster than the Selection Sort once it was running correctly. I struggled to understand the recursive nature and how the base case could be satisfied with the highIndex being greater than the lowIndex. Looking back through the animations in zybooks was incredibly helpful in understanding both algorithms. The quicksort algorithm is faster because it sorts as it iterates through each value. By determining the middle of the data set and sorting everything around whatever that value was, items are put on the correct side of the “pivot” value before being split into two separate groups. This is just repeated until each group contained only one or no value within it. Making the flowchart to show the recursive nature of the algorithm was harder than writing the actual code, because I wanted to make it easier to understand.