Algorithmic Sorting

Adam Babbit

Anton Kiselev

**Introduction**

Having a list of sorted data points is a valuable part of data manipulation. It allows for more complex algorithms to navigate the data with ease. These processes that sort the data can range from super lightweight and quick processes to taxing and long procedures. The length and complexity of these sorting algorithms are defined by their Big-O complexity. The Big-O complexity is a general way for describing the worst-case scenario based on time required to run the algorithm. They are general used to determine the time it will take for an algorithm to run as the size of the data set increases. This report will analyze the process used by the authors to find, and test the complexity of a bubble sort, and a quick sort algorithm.

**Complexity-Analysis**

The Big-O complexity of an algorithm is found based on how it goes through the length of the array. The bubble sort algorithm has a Big-O complexity of because it goes through the length of the array once per each element of the n. As you can see in Figure 1, for each element another for loop is run that also goes through the length again

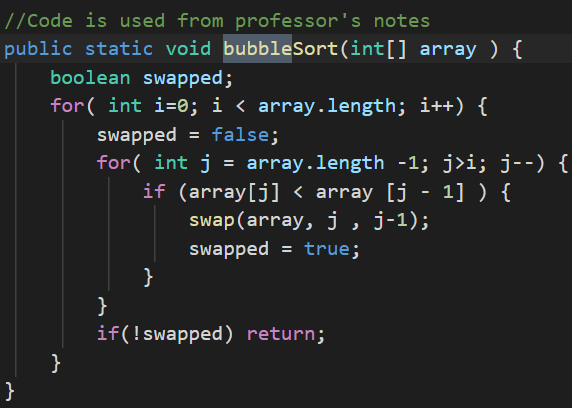


Figure : Bubble Sort Algorithm

The quicksort’s algorithmic complexity is a bit more advanced than the bubble sort. With that being said, it’s Big-O complexity is also a meaning that the worst case scenario is also n2. However the average complexity, Big-, is less. The average of bubble sort is while the quick sort has a Big-Theata of .

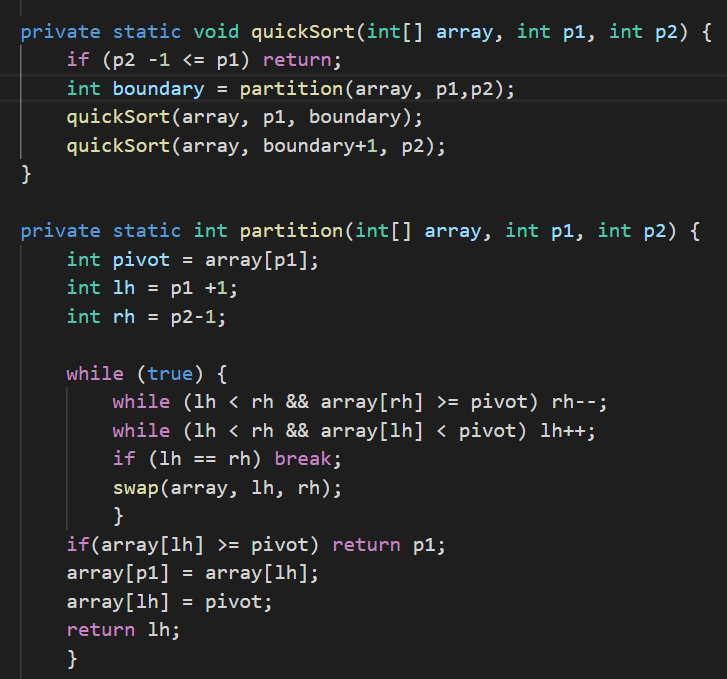


Figure : Quick Sort

This makes quicksort a good algorithm for use as it has a better average time complexity and no worse worst-case time complexity.

**Experimental Design**