Tristan Morse

Professor Lee Johnson

CSCI 202: Data Structures

17 March 2017

Homework 5 Chapter 18 Explanation

After testing these sorting algorithms, I came up with some interesting results. First off was the Selection sort which “sorts elements by repetitively finding a particular element and putting it in its final position” (Java Foundations 4th Edition). In a random list, it had made the lowest amount of comparisons in relation to the other methods with the fourth fastest speed. In an ordered list, it made no comparisons and took the longest.

Next was the Insertion sort which “sorts elements by repetitively inserting a particular element into a previously sorted sublist” (Java Foundations 4th Edition). In a random list, it had the second highest amount of comparisons and was the third fastest. In an ordered list, there are no comparisons done and completed in the quickest time.

After that is the Bubble sort which “sorts elements by repeatedly comparing adjacent values and swapping them” (Java Foundations 4th Edition). In a random list, it took the longest amount of time by a longshot and had the most comparisons. In an ordered list, it had the same result.

Also, there is the Quicksort which “sorts elements by partitioning the unsorted elements into two partitions and then recursively sorting each partition” (Java Foundations 4th Edition). In a random list, it had the quickest time and had the second smallest amount of comparisons. In an ordered list, it had the second fastest speed with the third least amount of comparisons.

Finally, there is the Merge sort which “sorts elements by recursively dividing the list in half until each sublist has one element and then merges the sublists” (Java Foundations 4th Edition). In a random list, it had the second fastest time and the third least amount of comparisons. In an ordered list, it had the fourth least amount of comparisons and third longest speed.

If I were to pick one of these sorts to use in order to sort out my data, I would more than likely choose the Quicksort, as it seems to have a balance of comparison numbers and speeds. This applies to both random lists as well as ordered lists, making it my top pick as a nicely balanced sorting algorithm.