Daniel Racz

CSC331H

Sorting Analysis

4/29/19

Summary of Sorting Analysis

Best/Worst Case Scenario Table:

Sort	Best Case Scenario	Worst Case Scenario
MergeSort	O(logn) – Any Assortment	O(logn) – Any Assortment
Selection	O(n2) – Any Assortment	O(n2) – Any Assortment
Insertion	O(n) – AlmostSorted	O(n2) – AlmostReversed
QuickSort	O(n*logn) – AlmostSorted	O(n2) - RandomAll
HeapSort	O(n*logn) – AlmostReversed	O(n*logn) - AlmostSorted

The five sorting algorithms used above were analyzed in a series of tests to examine the number of swaps and comparisons they each require. I created a sorting program that would ask the user to input either an option of 100 or 1000 integers and to select a scenario of how they will be given before sorting. Then five sorting algorithms are shown to be used and the program will output the unsorted and sorted lists with a swap and comparison insight once the sorting has been done.

Given the scenarios that are listed in the tables below, the hypothesis I came up before the trials started was that MergeSort would be able to deconstruct any situation consistently and all the other algorithms would vary and conclude in much higher numbers than MergeSort overall. The only time MergeSort falls short is against QuickSort in terms of its swaps and comparisons ration. MergeSort is on a 680 swap/680 comparison level and QuickSort can achieve an average of 628 swaps/630 comparisons on all scenarios. But this is only in the realm of testing 100 element arrays. MergeSort remains victorious with a consistent 9,987 swaps/comparisons vs. numbers that reach 10 to 100 thousand on other sorting algorithms. Quicksort falls short once the array becomes larger and larger. The times that MergeSort may fall is when memory may be in question. The algorithm requires a temporary array that is the size of the original array to be created to transfer over the sorted data. When introducing Heapsort into the mix, it's a bit slower out of the other two but still more effective than the two below which are Selection and Insertion.

Selection and Insertion sort are at the bottom of the list in terms of efficiency for sorting for my test runs. Selection Sort is consistent with all cases since it is comparing each value from the

index. Insertion is better with the array already sorted but will not perform as good when the array is in descending order as it compares and shifts each value it goes through. On average cases these may be equal in terms of efficiency but not the top choices compared to the rest of the list of algorithms.

In the end, big O notation is helpful to be able to analyze the data given and the algorithms we have at hand. It will help select the best method for that particular instance so that speed is always efficient.

**Numbers below were calculated on an average of three trials per experiment.*

-RandomALL -Completely Random Numbers

-RandomLast – Almost sorted in ascending order(90% in increasing order,10% random)

-AlmostSorted – Array is sorted except last 10%

AlmostReversed – Almost sorted in descending order (10% are random)

Table 1: Measuring Array of 100 Elements

Sort Method	RandomAll	RandomLast	AlmostSorted	AlmostReversed
MergeSort	680 swaps 680 comparisons	680 swaps 680 comparisons	680 swaps 680 comparisons	680 swaps 680 comparisons
Selection	99 swaps 4950 comparisons	99 swaps 4950 comparisons	99 swaps 4950 comparisons	99 swaps 4950 comparisons
Insertion	2464 swaps 2563 comparisons	461 swaps 560 comparisons	337 swaps 436 comparisons	4572 swaps 4670 comparisons
QuickSort	513 swaps 513 comparisons	497 swaps 497 comparisons	679 swaps 689 comparisons	823 swaps 823 comparisons
HeapSort	582 swaps 1158 comparisons	631 swaps 1297 comparisons	630 swaps 1291 comparisons	519 swaps 998 comparisons

Table 2: Measuring Array of 1000 Elements

Sort Method	RandomAll	RandomLast	AlmostSorted	AlmostReversed
MergeSort	9,987 swaps 9987 comparisons	9,987 swaps 9987 comparisons	9,987 swaps 9987 comparisons	9,987 swaps 9987 comparisons
Selection	999 swaps 499,500 comparisons	999 swaps 499,500 comparisons	999 swaps 499,500 comparisons	999 swaps 499,500 comparisons
Insertion	246,611 swaps 247,610 comparisons	88,248 swaps 89,249 comparisons	31,939 swaps 32,938 comparisons	470,089 swaps 471,088 comparisons
QuickSort	10,686 swaps 10,686 comparisons	39,404 swaps 39,404 comparisons	6,678 swaps 6,678 comparisons	11,501 swaps 11,501 comparisons
HeapSort	9,024 swaps 19,545 comparisons	9,439 swaps 20,909 comparisons	9,607 swaps 21,369 comparisons	8,455 swaps 18,075 comparisons