Assignment 4

Charles Denney

U9676-2161

Jashen Sambon

U2010-8775

Group 15

**Processor:** Intel(R) Core(TM) i5-9400F CPU 6 Core(s), 6 Logical Processor(s)

**Processor Speed:** 2.90GHz, 2904 Mhz

**Memory:** 8.00 GB

**Operating System:** Windows 10 Home

**Part 1: Sort**

**Algorithms Set I**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Timings of Sorting Algorithms on 10,000 items (seconds) | | | | |
| Sorting Algorithm | Initial Ordering of Items | | | |
| Descending | Ascending | Random Order 1 | Random Order 2 |
| bubbleSort.py | 7.2 | 2.7 | 5.3 | 5.4 |
| bubbleSortB.py | 7.1 | 0.001 | 5.3 | 5.3 |
| insertionSort.py | 4.8 | 0.001 | 2.5 | 2.5 |
| selectionSort.py | 2.1 | 2.2 | 2.0 | 2.0 |
| mergeSort.py | 0.02 | 0.02 | 0.03 | 0.03 |
| quicksort.py | 0.01 | 0.01 | 0.01 | 0.02 |

**Part 2: Sort Algorithms Set II Algorithms**

**Merge Sort**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Timings of Merge Sort (Seconds) | | | | |
| List Size | Initial Ordering of Items | | | |
| Descending | Ascending | Random Order 1 | Random Order 2 |
| 100,000 | 0.3 | 0.3 | 0.4 | 0.4 |
| 200,000 | 0.6 | 0.6 | 0.8 | 0.8 |
| 400,000 | 1.2 | 1.2 | 1.8 | 1.8 |

**Quick Sort**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Timings of Merge Sort (Seconds) | | | | |
| List Size | Initial Ordering of Items | | | |
| Descending | Ascending | Random Order 1 | Random Order 2 |
| 100,000 | 0.2 | 0.1 | 0.2 | 0.2 |
| 200,000 | 0.3 | 0.3 | 0.5 | 0.4 |
| 400,000 | 0.7 | 0.6 | 0.9 | 0.9 |

**Part 3: Questions**

1. For bubble sort algorithms, if an array is in descending order, it requires each element to be swapped in comparison to an array that is in ascending order, it only requires n-1 passes to be swapped which is less complex.
2. Bubble sort B algorithm takes a lot less time with exchanging elements within an ascending array because the algorithm stops early if no element is swapped or exchanged. In comparison to regular bubble sort algorithms that reiterates through the algorithm even if an element is no swapped.
3. Insertion sort algorithms takes a lot less time on ascending arrays because it requires only one comparison “O(N)”, where as descending arrays may require examinations of all elements within the sorted part of the array “O(N^2)”.
4. Bubble sort algorithms that have an array in descending order requires a comparison of each element next to the current iterated element, where as insertion sort only requires a comparison of the current iterated element to another element within the sorted part of the array.
5. Selection sort does not check sorted elements and insertion sort iterates all elements that are already sorted. That is why selection sort algorithm takes less time on a descending ordered list than a insertion sort algorithm’s list.
6. The reason why quick sort is the fastest advanced sort on random items is because it compares multiple values within each iteration, thus requiring less iterations than merge sort.