**Bubble Sort**

|  |  |
| --- | --- |
| **Number of Elements (Random Input)** | **Time (Micro Seconds)** |
| 10 | 1.0 |
| 100 | 129.33 |
| 1000 | 1367.00 |
| 10000 | 110424.00 |

|  |  |
| --- | --- |
| **Number of Elements (Nearly Sorted Input)** | **Time (Micro Seconds)** |
| 10 | 0.67 |
| 100 | 82.67 |
| 1000 | 1129.67 |
| 10000 | 74159.67 |

**Insertion Sort**

|  |  |
| --- | --- |
| **Number of Elements (Nearly Sorted Input)** | **Time(Micro Seconds)** |
| 10 | < 0.0 (unable to display accurate time) |
| 100 | 7.67 |
| 1000 | 323.33 |
| 10000 | 3069.67 |

|  |  |
| --- | --- |
| **Number of Elements (Random Input)** | **Time(Micro Seconds)** |
| 10 | 0.33 |
| 100 | 49.00 |
| 1000 | 991 |
| 10000 | 8077.67 |

**Selection Sort**

|  |  |
| --- | --- |
| **Number of Elements (Random Input)** | **Time (Micro Seconds)** |
| 10 | 1.00 |
| 100 | 66.00 |
| 1000 | 770.0 |
| 10000 | 35239.33 |

|  |  |
| --- | --- |
| **Number of Elements (Nearly Sorted Input)** | **Time (Micro Seconds)** |
| 10 | 0.33 |
| 100 | 67.67 |
| 1000 | 1154.67 |
| 10000 | 33732.00 |

In terms of time complexity, all three of the sorts had significant increase in time taken to sort as the data size increased confirming its O(N^2) time complexity. Insertion Sort was the quickest followed by Selection Sort, and then Bubble Sort, this might be due to the amount of times element were swapped around. When a data set that is partially sorted is used, both Bubble Sort and Insertion Sort showed significant improvement in their times while Selection Sort showed similar results compared to the random data set. In terms of its other characteristics, all three sorts are in-place sorting, due to using only one temporary variable for swapping. Only both insertion and bubble sort are stable sorts while selection is not.