**CS590 – Algorithms Assignment 1**

**Name: Sushant Bhat**

**CWID: 10474365**

**Abstract:**

This report discusses and provides findings on 3 algorithms that have been used to sort an array of integer vectors namely Insertion Sort whose naive implementation is already given, an improved version of the same Insertion Sort and a Merge Sort algorithm

The problem statement is as follows: Given a function that computes the length of vector *v* with dimension *n* as , we have to precompute the length of the given vector and implement an efficient Insertion Sort and a Merge sort algorithm.

**Algorithms:**

Three algorithms have been implemented for this assignment:

1. **Naive Insertion Sort:**

Insertion Sort is an efficient algorithm at sorting a small dataset. While it is not the best or the most efficient sorting algorithm, it is relatively easier to implement than many other sorting algorithms and is best when the data is almost or partially sorted.

1. **Improved Insertion Sort:**

As opposed to the inefficient Insertion Sort algorithm above, the length of the array of integers is precomputed beforehand which significantly reduces the running time of the algorithm.

1. **Merge Sort:**

This is a divide and conquer algorithm and is far efficient than any of the two algorithms stated above when it comes to larger data sets. This algorithm divides the data into n subsets and sorts the subsets and merges it after the operation.

**Results:**

Following are the running times and graph plots of each of the 3 algorithms in random, sorted and reverse sorted cases:

**Insertion Sort (Naive):**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dimension** | **n = 10** | | | **n = 25** | | | **n = 50** | | |
| **Size(m)** | Random Vector | Sorted Vector | Inverse Sorted Vector | Random Vector | Sorted Vector | Inverse Sorted Vector | Random Vector | Sorted Vector | Inverse Sorted Vector |
| 1000 | 9 | 22 | 32 | 22 | 1 | 123 | 45 | 1 | 189 |
| 2500 | 54 | 0 | 210 | 138 | 1 | 532 | 300 | 1 | 886 |
| 5000 | 213 | 1 | 1236 | 559 | 1 | 2263 | 1272 | 2 | 4865 |
| 10000 | 862 | 1 | 1926 | 2324 | 1 | 5112 | 5450 | 2 | 10653 |
| 25000 | 6270 | 1 | 11535 | 15062 | 3 | 15062 | 24360 | 1 | 45362 |
| 50000 | 24360 | 1 | 45362 | 84556 | 5 | 84556 | 188730 | 9 | 188730 |
| 100000 | 142890 | 4 | 206535 | NA | 12 | NA | NA | 25 | NA |
| 250000 | NA | 9 | NA | NA | 24 | NA | NA | 50 | NA |

**Improved Insertion Sort:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dimension** | **n = 10** | | | **n = 25** | | | **n = 50** | | |
| **Size(m)** | Random Vector | Sorted Vector | Inverse Sorted Vector | Random Vector | Sorted Vector | Inverse Sorted Vector | Random Vector | Sorted Vector | Inverse Sorted Vector |
| 1000 | 1 | 0 | 1 | 2 | 0 | 2 | 2 | 0 | 2 |
| 2500 | 6 | 0 | 13 | 8 | 0 | 16 | 7 | 0 | 16 |
| 5000 | 83 | 1 | 136 | 93 | 0 | 121 | 89 | 1 | 121 |
| 10000 | 193 | 1 | 325 | 189 | 1 | 334 | 201 | 1 | 343 |
| 25000 | 899 | 1 | 1799 | 955 | 3 | 1869 | 969 | 3 | 1861 |
| 50000 | 1735 | 1 | 3598 | 1833 | 3 | 3587 | 1798 | 8 | 3610 |
| 100000 | 11998 | 4 | 21103 | 12389 | 12 | 19770 | 12449 | 25 | 20884 |
| 250000 | 44528 | 9 | 88920 | 55548 | 79 | 109672 | 57585 | 99 | 109832 |

**Merge Sort:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dimension** | **n = 10** | | | **n = 25** | | | **n = 50** | | |
| **Size(m)** | Random Vector | Sorted Vector | Inverse Sorted Vector | Random Vector | Sorted Vector | Inverse Sorted Vector | Random Vector | Sorted Vector | Inverse Sorted Vector |
| 1000 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 |
| 2500 | 3 | 2 | 3 | 3 | 4 | 3 | 6 | 6 | 6 |
| 5000 | 6 | 4 | 4 | 6 | 6 | 7 | 9 | 8 | 9 |
| 10000 | 13 | 11 | 13 | 17 | 17 | 16 | 36 | 36 | 35 |
| 25000 | 66 | 76 | 71 | 99 | 93 | 99 | 111 | 112 | 111 |
| 50000 | 121 | 111 | 121 | 151 | 153 | 143 | 178 | 181 | 183 |
| 100000 | 187 | 194 | 189 | 232 | 237 | 227 | 333 | 322 | 328 |
| 250000 | 433 | 420 | 391 | 553 | 562 | 559 | 641 | 736 | 751 |

**Conclusion:**

With a worst-case complexity of O(nlogn) Merge Sort easily outperforms Insertion sort as well as Improved Insertion sort when large data sets are present. When it come to the data which is already sorted, both versions of Insertion Sorts are highly efficient but even so there isn’t much difference even in the runtime of Merge Sort. For reverse sorted arrays, Merge Sort is the best algorithm when compared to the other two.