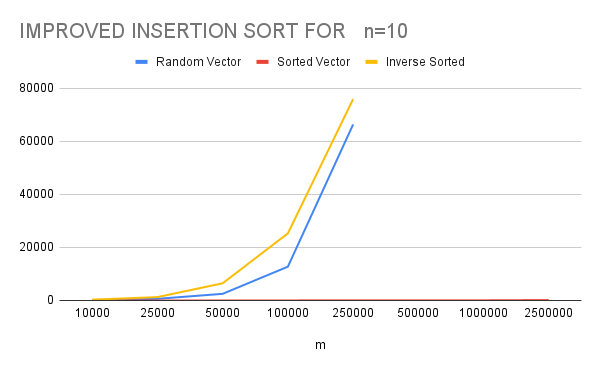
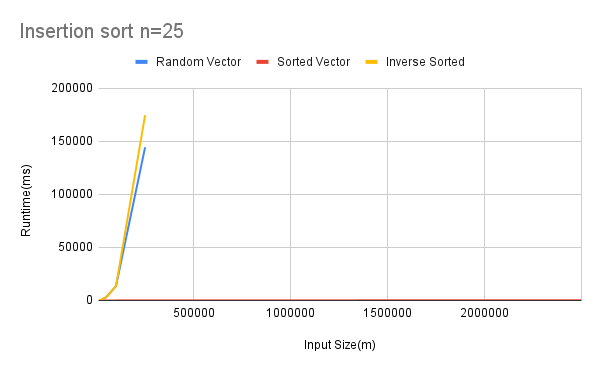
CS 590 Assignment 1 – Yash Bharambay

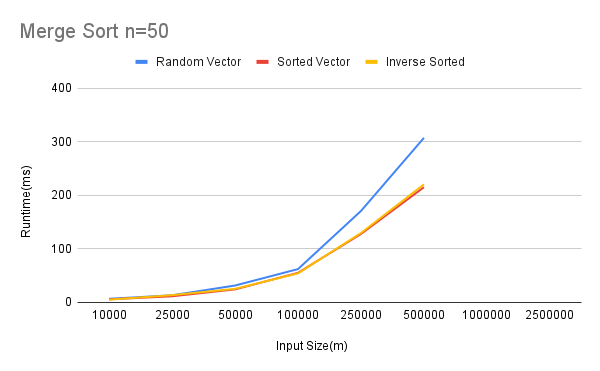
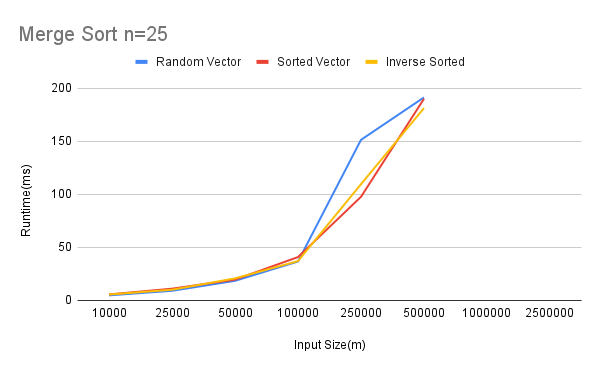
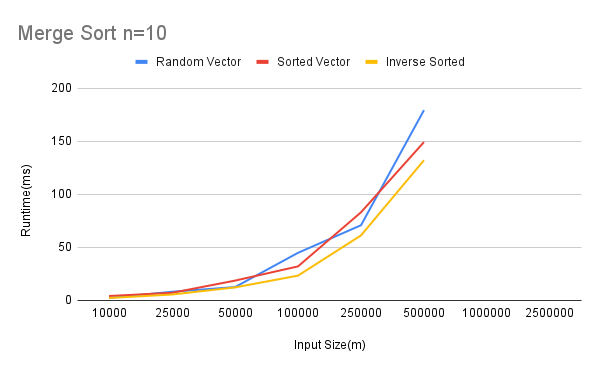
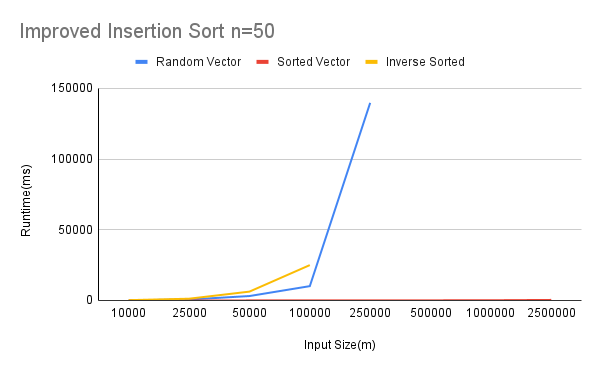
# Abstract:

In this report we will be analyzing three algorithms. Naive Insertion sort, Improved insertion sort and Merge sort.

**Graphs:**







**TABLES** -

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Insertion sort** | | | | | | | | | |
|  | **n=10** | | | **n=25** | | | **n=50** | | |
|  |  |  |  |  |  |  |  |  |  |
| m | Random | Sorted | Inverse Sorted | Random Vector | Sorted Vector | Inverse Sorted | Random Vector | Sorted Vector | Inverse Sorted |
| 10000 | 156 | 1 | 298 | 102 | 1 | 101 | 102 | 2 | 208 |
| 25000 | 614 | 1 | 1264 | 629 | 3 | 825 | 654 | 6 | 1232 |
| 50000 | 2492 | 3 | 6461 | 3133 | 6 | 3073 | 3114 | 9 | 6245 |
| 100000 | 12736 | 3 | 25292 | 13551 | 12 | 13524 | 10117 | 15 | 24989 |
| 250000 | 66353 | 13 | 75923 | 144375 | 30 | 174655 | 139713 | 44 |  |
| 500000 |  | 18 |  |  | 53 |  |  | 87 |  |
| 1000000 |  | 44 |  |  | 99 |  |  | 135 |  |
| 2500000 |  | 136 |  |  | 233 |  |  | 368 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Improved Insertion Sort** | | | | |
|  |  |  |  |  |
| **n** | **order** | **m** | **t(ms)** | **t(ms) (average)** |
| 10 | random | 10000 | [ 107, 98, 86, 85, 76, 79, 78, 94, 104 ] | 82.6 |
| 10 | sorted | 10000 | [1,0,1,1,0,1,1,1,0,1] | 0.7 |
| 10 | inverse | 10000 | [319,209,286,246,329,257,239,270,210,256] | 262.1 |
| 25 | random | 10000 | [135, 87, 84, 85, 87, 84, 85, 86, 86, 84, 84] | 98.7 |
| 25 | sorted | 10000 | [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1] | 1 |
| 25 | inverse | 10000 |  |  |
| 50 | random | 10000 | [ 93, 87, 86, 85, 86, 99, 86, 91, 88, 86, 88] | 97.5 |
| 50 | sorted | 10000 | [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1] | 1 |
| 50 | inverse | 10000 |  |  |
|  |  |  |  |  |
| **n** | **order** | **m** | **t(ms)** | **t(ms) (average)** |
| 10 | random | 25000 | [ 576, 556, 545, 565, 532, 538, 571, 537, 528, 530, 538 ] | 601.6 |
| 10 | sorted | 25000 | [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1] | 1 |
| 10 | inverse | 25000 |  |  |
| 25 | random | 25000 | [840, 669, 667, 675, 668, 686, 666, 688, 667, 669, 670] | 756.5 |
| 25 | sorted | 25000 | [2, 2, 2, 2, 2, 2, 2, 3, 3, 2, 2 ] | 2.4 |
| 25 | inverse | 25000 |  |  |
| 50 | random | 25000 | [534, 557, 555, 527, 525, 601, 933, 757, 596, 549, 529] | 666.3 |
| 50 | sorted | 25000 | [5, 5, 5, 5, 4, 4, 5, 4, 5, 4, 4, ] | 5 |
| 50 | inverse | 25000 |  |  |
|  |  |  |  |  |
| **n** | **order** | **m** | **t(ms)** | **t(ms) (average)** |
| 10 | random | 50000 | [2815, 2875, 2699, 2666, 2646, 2750, 2667, 2657, 2663, 2644, 2824] | 2990.6 |
| 10 | sorted | 50000 | [2, 3, 2, 2, 3, 3, 2, 3, 3, 2, 2] | 2.7 |
| 10 | inverse | 50000 |  |  |
| 25 | random | 50000 | [2665, 2664, 2666, 2674, 2676, 2795, 2660, 2731, 2728, 2661, 2758] | 2967.8 |
| 25 | sorted | 50000 | [5, 5, 3, 4, 4, 4, 4, 4, 5, 5, 4] | 4.7 |
| 25 | inverse | 50000 |  |  |
| 50 | random | 50000 | [2846, 2786, 2736, 2736, 2717, 2731, 2732, 2746, 2686, 2719, 2693] | 3012.8 |
| 50 | sorted | 50000 | [5, 5, 3, 4, 4, 4, 4, 4, 5, 5, 4] | 8.4 |
| 50 | inverse | 50000 |  |  |
|  |  |  |  |  |
| **n** | **order** | **m** | **t(ms)** | **t(ms) (average)** |
| 10 | random | 100000 | [21959, 20655, 10795, 10963, 10770, 29731, 42995, 51900, 49614, 47386, 48524] | 32429.2 |
| 10 | sorted | 100000 | [4, 3, 3, 4, 3, 4, 4, 3, 4, 4, 3] | 3.9 |
| 10 | inverse | 100000 |  |  |
| 25 | random | 100000 | [10743, 10706, 10659, 10692, 11370, 30088, 44701, 48197, 51900, 47978, 50799] | 32783.3 |
| 25 | sorted | 100000 | [10, 10, 10, 9, 9, 9, 9, 9, 10, 10, 10] | 10.5 |
| 25 | inverse | 100000 |  |  |
| 50 | random | 100000 | [10909, 10913, 10705, 10842, 10786, 30255, 53593, 49397, 51282, 52889, 47996] | 42194.4 |
| 50 | sorted | 100000 | [17, 17, 17, 17, 18, 17, 16, 18, 17, 17, 17] | 18.8 |
| 50 | inverse | 100000 |  |  |
|  |  |  |  |  |
| **n** | **order** | **m** | **t(ms)** | **t(ms) (average)** |
| 10 | random | 250000 | [51448 , ] |  |
| 10 | sorted | 250000 | [8, 8, 7, 7, 7, 8, 7, 7, 7, 8, 8] | 8.2 |
| 10 | inverse | 250000 |  |  |
| 25 | random | 250000 |  |  |
| 25 | sorted | 250000 | [19, 19, 22, 19, 19, 19, 20, 19, 21, 19, 19] | 20.9 |
| 25 | inverse | 250000 |  |  |
| 50 | random | 250000 |  |  |
| 50 | sorted | 250000 | [44, 45, 42, 42, 47, 43, 42, 47, 42, 42, 43] | 46.7 |
| 50 | inverse | 250000 |  |  |
|  |  |  |  |  |
| **n** | **order** | **m** | **t(ms)** | **t(ms) (average)** |
| 10 | random | 500000 |  |  |
| 10 | sorted | 500000 | [19, 19, 19, 19, 19, 19, 18, 18, 19, 18, 18] | 18.5 |
| 10 | inverse | 500000 |  |  |
| 25 | random | 500000 |  |  |
| 25 | sorted | 500000 | [49, 51, 49, 49, 48, 48, 48, 48, 48, 49, 48] | 49 |
| 25 | inverse | 500000 |  |  |
| 50 | random | 500000 |  |  |
| 50 | sorted | 500000 | [68, 66, 66, 66, 66, 67, 66, 67, 66, 66, 66] | 66.5 |
| 50 | inverse | 500000 |  |  |
|  |  |  |  |  |
| **n** | **order** | **m** | **t(ms)** | **t(ms) (average)** |
| 10 | random | 1000000 |  |  |
| 10 | sorted | 1000000 | [47,44,41,44,49,51,42,41,47,44] | 46.6 |
| 10 | inverse | 1000000 |  |  |
| 25 | random | 1000000 |  |  |
| 25 | sorted | 1000000 | [96, 96, 95, 95, 96, 100, 95, 96, 95, 96, 97] | 99 |
| 25 | inverse | 1000000 |  |  |
| 50 | random | 1000000 |  |  |
| 50 | sorted | 1000000 | [204, 171, 175, 173, 169, 172, 168, 168, 168, 236, 189] | 199.3 |
| 50 | inverse | 1000000 |  |  |
|  |  |  |  |  |
| **n** | **order** | **m** | **t(ms)** | **t(ms) (average)** |
| 10 | random | 2500000 |  |  |
| 10 | sorted | 2500000 | [119, 143, 133, 155, 137, 149, 128, 91, 156, 121, 114] | 137.3 |
| 10 | inverse | 2500000 |  |  |
| 25 | random | 2500000 |  |  |
| 25 | sorted | 2500000 | [192, 207,217, 271, 264, 227, 191, 217, 290, 289, 221] | 233 |
| 25 | inverse | 2500000 |  |  |
| 50 | random | 2500000 |  |  |
| 50 | sorted | 2500000 | [301,371,384,321,345,367,368,311,371,355] | 368 |
| 50 | inverse | 2500000 |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Merge Sort | | | | | | | | | | |
|  | **10** | | | **25** | | | **50** | | |
| m | random | sorted | inverse | random | sorted | inverse | random | sorted | inverse |
| 10000 | 2.4 | 4.2 | 2.3 | 4.8 | 5.7 | 5.6 | 6.8 | 5.9 | 5.2 |
| 25000 | 8.2 | 7.2 | 5.6 | 9.1 | 11.1 | 10.1 | 13.4 | 11.5 | 13.6 |
| 50000 | 12.6 | 18.7 | 12.3 | 18.5 | 19.6 | 20.8 | 31.6 | 24.4 | 25.4 |
| 100000 | 44.9 | 32.1 | 23.4 | 36.7 | 40.9 | 37.1 | 62.3 | 55.3 | 54.2 |
| 250000 | 70.9 | 83.1 | 61.3 | 151.4 | 97.7 | 109.6 | 170.9 | 127.9 | 129.7 |
| 500000 | 179.4 | 149.3 | 132.2 | 191.4 | 190.1 | 181.3 | 307.5 | 215.3 | 220.6 |
| 1000000 |  |  |  |  |  |  |  |  |  |
| 2500000 |  |  |  |  |  |  |  |  |  |

## Improved Insertion sort:

m = 10000 n = 10

|  |  |  |
| --- | --- | --- |
| Random | Sorted | Inverse |
| 82.6 | 0.7 | 262 |

## Merge sort:

m = 10000 n = 10

|  |  |  |
| --- | --- | --- |
| Random | Sorted | Inverse |
| 2.4ms | 4.2ms | 2.3ms |

**Discussion**:

Every time we needed to compare the length of two vectors in naïve insertion sort, we calculated the vector's length. I made an array and put the length of each vector in the array in the Improved Insertion Sort and Merge Sort application. I then used this array to sort the vector array. I swapped the two elements in the vector length array whenever I had to swap two elements in the array of vectors. This was done to make sure that the vector length array's vector length index corresponded to the vector's length .As shown in the tables and the graphs, changing the code in this manner significantly reduced the time complexity.

**Result** :

The sorting algorithms were analyzed . We determined that pre-calculating the vector lengths for insertion sort and merge sort allows us to quickly sort the array of vectors.