Test Results for the implementation of the algorithms

```
Original, unsorted values of test1.txt :

029 081 089 050 038 060 091 088 064 035
052 090 065 061 009 065 053 017 074 004
014 040 047 071 064 078 095 006 037 064
026 007 060 046 043 099 002 076 033 013
080 067 089 065 026 012 062 064 019 012


Sorted Values of test1.txt using Insertiong Sort:

002 004 006 007 009 012 012 013 014 017
019 026 026 029 033 035 037 038 040 043
046 047 050 052 053 060 060 061 062 064
064 064 064 065 065 065 067 071 074 076
078 080 081 088 089 089 090 091 095 099

Original, unsorted values of test1.txt :

029 081 089 050 038 060 091 088 064 035
052 090 065 061 009 065 053 017 074 004
014 040 047 071 064 078 095 006 037 064
026 007 060 046 043 099 002 076 033 013
080 067 089 065 026 012 062 064 019 012


Sorted Values of test1.txt using Quicksort:

002 004 006 007 009 012 012 013 014 017
019 026 026 029 033 035 037 038 040 043
046 047 050 052 053 060 060 061 062 064
064 064 064 065 065 065 067 071 074 076
078 080 081 088 089 089 090 091 095 099

Original, unsorted values of test1.txt :

029 081 089 050 038 060 091 088 064 035
052 090 065 061 009 065 053 017 074 004
014 040 047 071 064 078 095 006 037 064
026 007 060 046 043 099 002 076 033 013
080 067 089 065 026 012 062 064 019 012


Sorted Values of test1.txt using Newsort:

002 004 006 007 009 012 012 013 014 017
019 026 026 029 033 035 037 038 040 043
046 047 050 052 053 060 060 061 062 064
064 064 064 065 065 065 067 071 074 076
078 080 081 088 089 089 090 091 095 099
```

Test Results for all six test files being sorted

```
Sorted Values of test1.txt:

Insertion sort comparison counter: 732

Quicksort comparison counter: 412

Newsort comparison counter: 2111


Sorted Values of test2.txt:

Insertion sort comparison counter: 781

Quicksort comparison counter: 970

Newsort comparison counter: 2791


Sorted Values of test3.txt:

Insertion sort comparison counter: 1580

Quicksort comparison counter: 1424

Newsort comparison counter: 3471


Sorted Values of test4.txt:

Insertion sort comparison counter: 244286

Quicksort comparison counter: 15767

Newsort comparison counter: 811677


Sorted Values of test5.txt:

Insertion sort comparison counter: 247902

Quicksort comparison counter: 239332

Newsort comparison counter: 1634011


Sorted Values of test6.txt:

Insertion sort comparison counter: 498357

Quicksort comparison counter: 256698

Newsort comparison counter: 1872702
```

Observations

1. From the test results that we get from running these three methods we see that the new sort algorithm fails to produce results that could be matched with the other two. Its comparison counter is the highest in every occasion and there is also a significant difference to its amount in regard to the others.

2. In the tests with 50 values as a max (small arrays) we notice that insertion sort and quicksort give us some mixed results where in two occasions one is performing better than the other but in another occasion however that is not the case and both algorithms perform similarly in terms of comparisons.

3. In the tests where the max number of values was set to 1000 (large arrays) we observe that quicksort outperformed all the other algorithms in all occasions. Nonetheless we still find one occasion where quicksort and insertion sort had similar results (test5.txt).

4. In Test4.txt we can clearly see a significant proficiency in performance from quicksort by completing the sorting with way less comparisons than the rest of the methods.

5. In Test6.txt it is noticeable that the quicksort algorithm completed the sorting with again a lot less comparisons that insertion sort, almost half. This goes to show the excellence of quicksort in sorting large arrays.