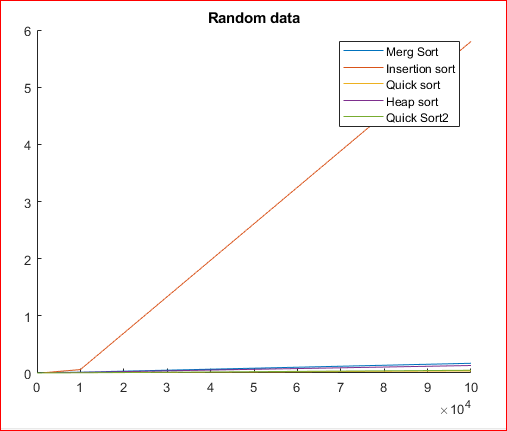
Git-Hub link for the code:

<https://github.com/ignitorkhan/Algorithim_Assignment2>

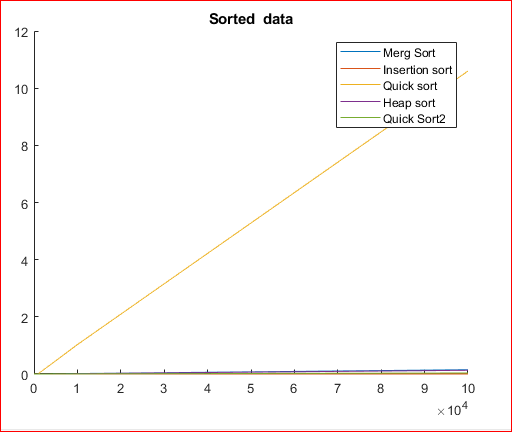
Question 1:

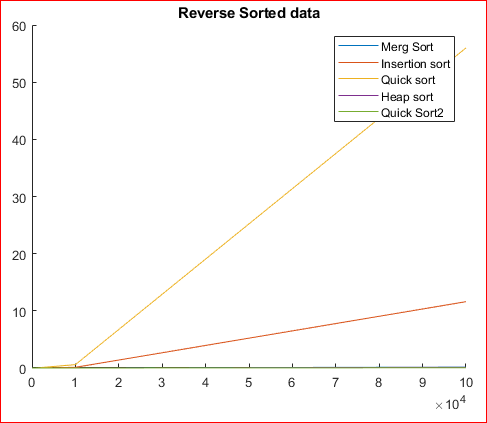
The results of different sorting algorithms are as follows. Due to the varied range of data the scale had to be chosen that accommodated the worst case making graphs appear closer to each other.

Random data:

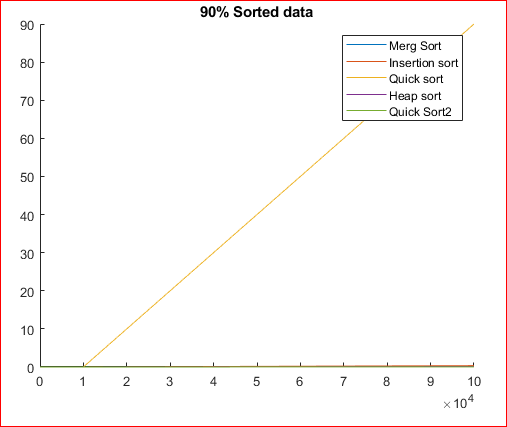


Sorted Data:



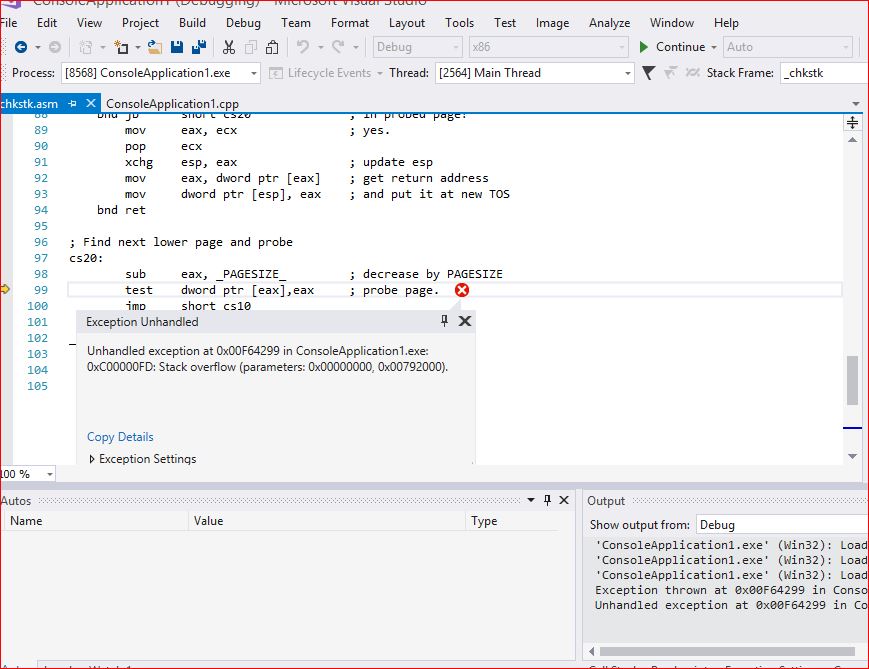
Reverse Sorted data:

90% sorted data:



**Note:**

1. At the size of 1000000 the recursive functions were unable to complete due to the stack overflow error. Therefore an iterative approach was used



1. While the data for 1000000 for insertion sort was not included because the taken was too long and its value skewed the result making the data illegible.
2. For smaller data sets the effective time taken was so small that it was recorded as 0.0 seconds by the computer.

**Question 2:**

The time complexity for merge sort is (nlogn) while the time complexity of insertion sort is (n^2).  
This means that the insertion sort algorithm does not scale well with the large amount of data although for small input size insertion sort is better.

At each call of the merge sort is algorithm we reduce the size of the arrays to be sorted.  
In the modified version of the algorithm, once the length of the divided array we stop the recursive calls and instead call the insertion sort on the smaller size arrays.  
Following are the empirically calculated ideal values of K for different sizes.

**Input size=1000**:

For input size of 1000. The mean time of different values of k is as follow

Merge sort:

Original merge sort takes time = 0.002 sec

Modified merge sort:

* K=100 time = 0.001 sec
* K=10 time = 0.001 sec
* K=1000 time = 0.0015 sec

So, k=1000 is threshold below which modified merge sort is faster.  
Insertion sort is always better.

**Input size=10,000**:

For input size of 10,000. The mean time of different values of k is as follow

Merge sort:

Original merge sort takes time = 0.016 sec

Modified merge sort:

* K=5000 time = 0.033 sec
* K=2500 time = 0.019 sec
* K=1250 time = 0.009 sec
* K=2000 time = 0.012 sec
* K=750 time = 0.006sec
* K=10000 time = 0.068 sec

So, k=2000 is threshold below which modified merge sort is faster

**Input size=50,000**:

For input size of 50,000. The mean time of different values of k is as follow

Merge sort:

Original merge sort takes time = 0.107 sec

Modified merge sort:

* K=50000 time = 1.63 sec
* K=25000 time = 0.86 sec
* K=12500 time = 0.412 sec
* K=6250 time = 0.210 sec
* K=3125 time = 0.110 sec
* K=1560 time = 0.03 sec

So, k=3125 is threshold below which modified merge sort is faster

**Input size=100,000**:

For input size of 100,000. The mean time of different values of k is as follow

Merge sort:

Original merge sort takes time = 0.21 sec

Modified merge sort:

* K=25000 time = 1.60 sec
* K=12500 time = 0.88 sec
* K=6250 time = 0.41 sec
* K=3125 time = 0.213 sec
* K= 1560 time = 0.07 sec

So, k=3125 is threshold below which modified merge sort is faster