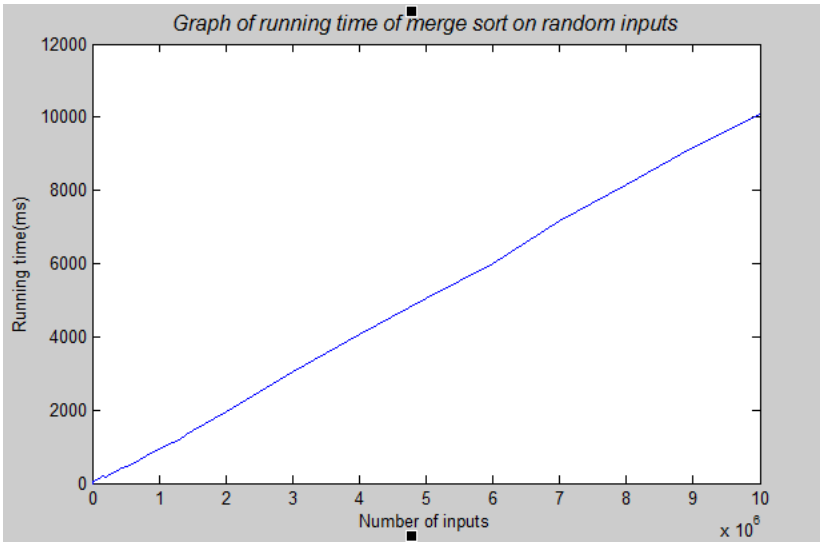
COMP20007 Design Of Algorithms
Assignment1_NonRecursive merge sort and quick sort

Hyeri(Tessa) Song 597952

Analyses of the Algorithms

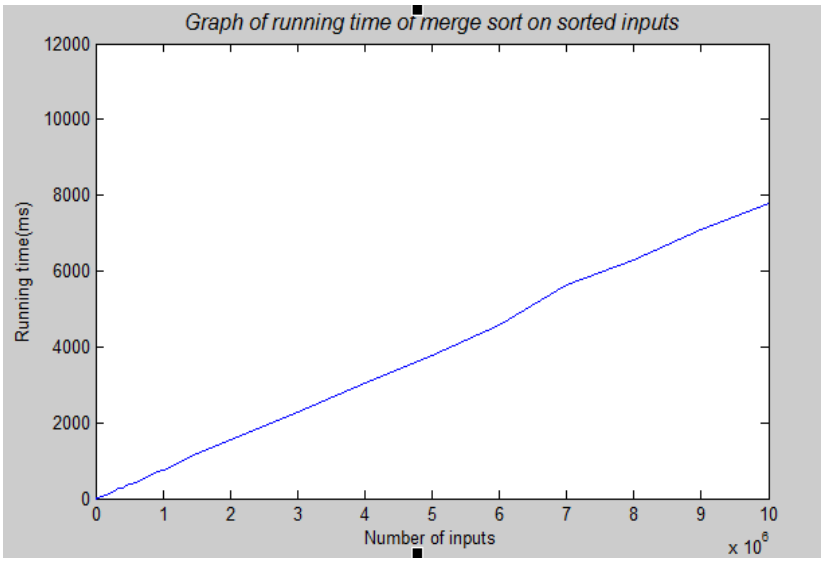
Both algorithms have been tested with three different kinds of inputs. One is randomly generated inputs, another is sorted inputs and the other is inputs where all the values are the same.

1) Merge sort

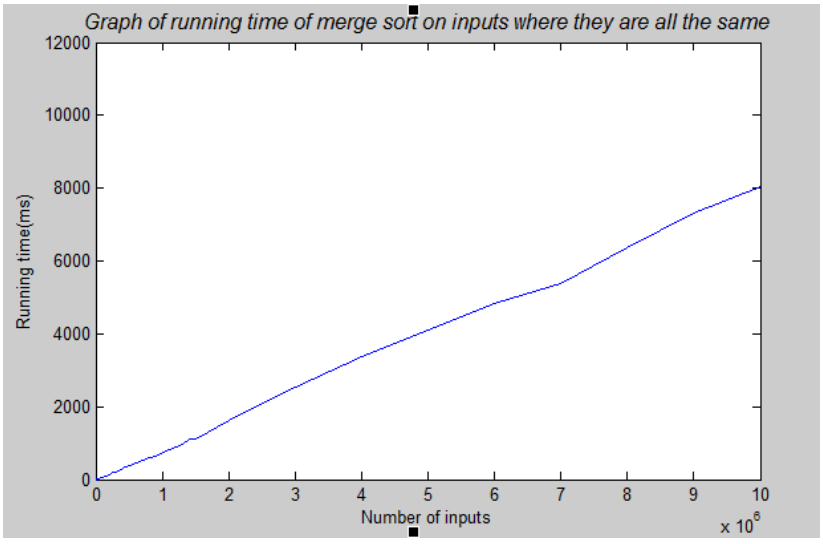


Generally, the Big O Running time of merge sort is O(nlogn). This is always the case on any input, even on the worst one. The upper bound of O(nlogn) is always guaranteed.

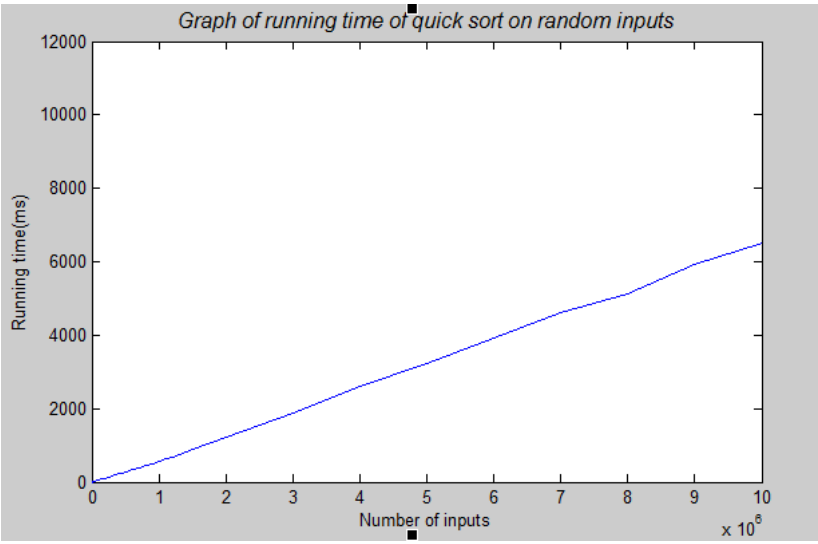
The first graph shows the running time of merge sort on random inputs and as you can see, the graph is increasing acceptably gradually representing O(nlogn). This is the case on the other two graphs as well.

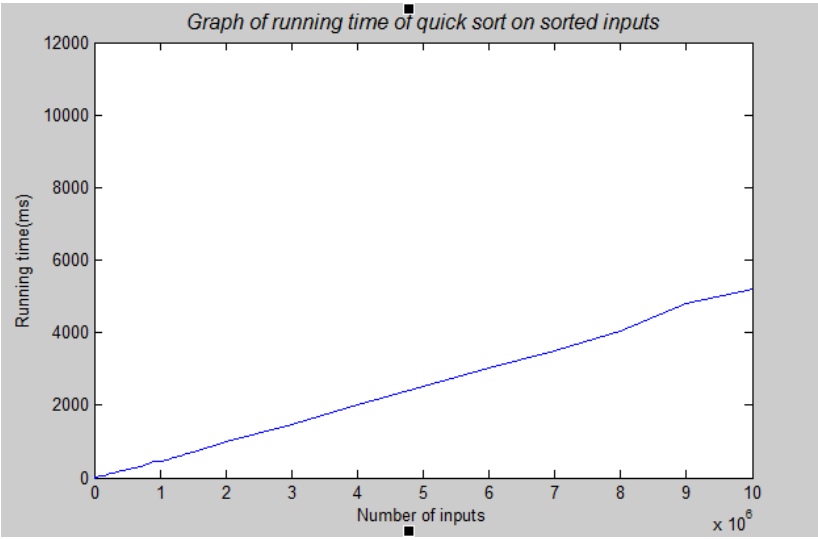


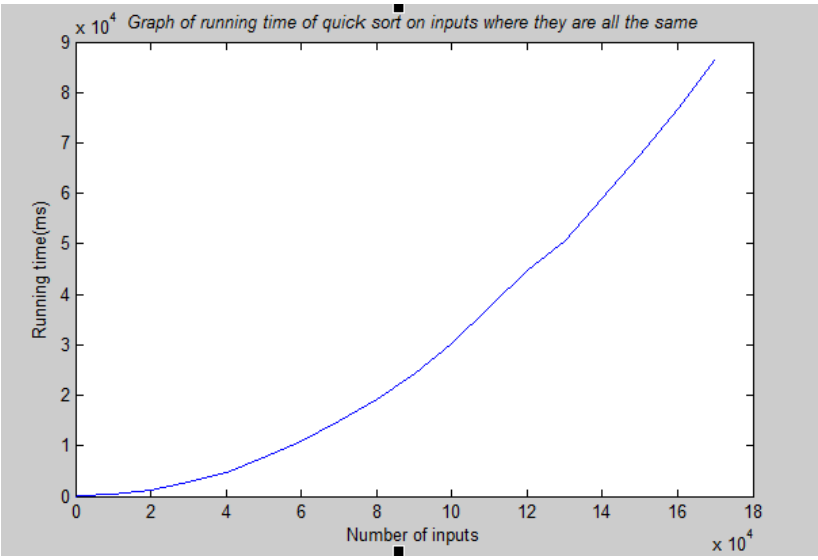
Basically, merge sort does not adhere to the type of input so that the three graphs can share the similar slope of curve.



2) Quick sort







When the efficiency of an algorithm is being evaluated, normally the input of worst case is considered representatively. However, it is neither helpful nor practical for quick sort since the possibility is extremely rare. The Big O running time of quick sort on worst case is O(n^2) but O(nlogn) , the average running time, is more likely to be considered as a practical running time.

Especially, the worst case can be easily avoided by choosing pivot values deliberately. In this program, in order to obtain a proper pivot value, three random elements are selected first and sorted and the median is taken as a pivot.

The first two graphs prove the practical running time of quick sort well. Like as those of merge sort above, the slopes are reasonably gradual so that increasing input size does not much strain the capability of quick sort.

The third graph shows a different shape of curve. This curve represents O(n^2) rather than O(nlogn). This happens when all the elements of the array are the same. However, this problem can be settled down without any difficulty by using Dutch flag partition so that we can divide the array strictly into three parts, smaller than pivot, same with pivot and greater than pivot. The running time of this case is just O(n) with this solution.

* Comparison between merge sort and quick sort.
Both sorting algorithms share the same Big O running time, especially, merge sort always can guarantee O(nlogn). However, according to the graphs above, quick sort is generally quicker than merge sort. One of the reason is that quick sort uses only O(1) space whereas merge sort uses O(n) space for temporary arrays to merge.