**READ ME**

**PART -I**

This package version includes Merge sort implementation code for the following flavors,

a) Allocate dynamic memory in each call to Merge for L and R

b) Use an auxiliary array B[ ] (same size as A). In each call to Merge, copy contents from A to B and then merge them back to A, in sorted order.

c) Use an auxiliary array B[ ] (same size as A). When data is in A[p..q..r], it is merged into B[p..r]. When data is in B[p..q..r], it is merged into A[p..r]

**Files Impacted:**

MergeSort.java

MergeSortAuxilary.java

MergeSortAlternateMerge.java

**ASSUMPTION:**  The first line of the input file is the number of input elements.

**PART - II**

Compare performance of Merge Sort and Quick Sort for sorting large amount of data that does not fit within memory. This is called external sorting. For more details, contact the instructor.

Package includes the regular implementation of the Quick sort.

**Files Impacted:**

QuickSort.java

Note: Performance tested and measured in cs1.utdallas.edu Unix box

**Common File:** SortAnalysis.java

**Steps to execute:**

* SortAnalysis Class acts as the executable main class to the sort methods implemented in other classes.
* It takes the input as file containing the numbers to be sorted and produces an output file containing the execution time for each sorting functions.
* The first line of the input file is assumed to be the number of input elements in the file to be sorted.