Ketan Kokane CSCI 661 Project 1 Report

Summary Report:

Compared the traditional quicksort with a variation of quicksort wherein for smaller subarrays are sorted using insertion sort.
For the partition function of both the quicksort algorithm used the first element as the pivot value. The sorting mechanism was executed on datasets of sizes ranging from 1K to 500K. Exhaustive reporting of the results can be found in the reports file.
It was noticed that for the dataset generated using Poisson distribution due to repetitive numbers the quick sort ends up doing N comparisons for every run and ends up with higher execution time.

For variation of the quicksort used the reference from [1] and used the techniques to reduce the number of subarrays on which the quicksort is to be used, so when the length of subarray is less than 10, then used normal insertion sort to for sorting the subarray.

To decide upon the size of the array which should be sorted using insertion, ran the code with different values ranging from 100 to 5, and found out that the best result was achieved using the length of 10.

Using the second algorithm the execution time decreased, as mentioned in the reference paper using a different sorting algorithm for small subarrays reduced the memory usage, however, did not exhaustively measure and report the memory usage improvement in this report.

[1] R.L. Wainwright, A Class of Sorting Algorithms based on Quicksort, Communications of the ACM, Vol. 28, No. 4, April 1985, pgs. 396-402