**605.202 Data Structures Lab 4**

This lab assignment requires you to compare the performance of two distinct sorting algorithms to obtain some appreciation for the parameters to be considered in selecting an appropriate sort. Write a **Quicksort** and a **Natural Merge Sort**. They should both be recursive or both be iterative, so that the overhead of recursion will not be a factor in your comparisons.

A **Quicksort** (or Partition Exchange Sort) divides the data into 2 partitions separated by a pivot. The first partition contains all the items which are smaller than the pivot. The remaining items are in the other partition. You will write four versions of Quicksort:

* Select the first item of the partition as the pivot. Treat partitions of size one and two as stopping cases.
* Same pivot selection. For a partition of size 100 or less, use an insertion sort to finish.
* Same pivot selection. For a partition of size 50 or less, use an insertion sort to finish.
* Select the median-of-three as the pivot. Treat partitions of size one and two as stopping cases.

As time permits consider examining additional, alternate methods of selecting the pivot for **Quicksort**.

**Merge** Sort is a useful sort to know if you are doing External Sorting. The need for this will increase as data sizes increase. The traditional **Merge** Sort requires double space. To eliminate this issue, you are to implement Natural **Merge** using a linked implementation. In your analysis be sure to compare to the effect of using a straight **Merge Sort** instead.

Create input files of four sizes: 50, 1000, 2000, 5000 and 10000 integers. For each size file make 3 versions. On the first use a randomly ordered data set. On the second use the integers in reverse order. On the third use the

integers in normal ascending order. (You may use a random number generator to create the randomly ordered file, but it is important to limit the duplicates to <1%. Alternatively, you may write a shuffle function to randomize one of your ordered files.) **This means you have an input set of 15 files plus whatever you deem necessary and reasonable.** Files are available in the Blackboard shell, if you want to copy them. Your data should be formatted so that each number is on a separate line with no leading blanks. There should be no blank lines in the file. Even though you are limiting the occurrence of duplicates, your sorts must be able to handle duplicate data.

Each sort must be run against all the input files. With five sorts and 15 input sets, you will have 75 required runs.

The size 50 files are for the purpose of showing the sorting is correct. Your code needs to print out the comparisons and exchanges (see below) and the sorted values. You must submit the input and output files for all orders of size 50, for all sorts. There should be 15 output files here.

The larger sizes of input are used to demonstrate the asymptotic cost. To demonstrate the asymptotic cost you will need to count comparisons and exchanges for each sort. For these files at the end of each run you need to print the number of comparisons and the number of exchanges but not the sorted data. It is to your advantage to add larger files or additional random files to the input - perhaps with 15-20% duplicates. You may find it interesting to time the runs, but this should be in addition to counting comparisons and exchanges.

Turn in an analysis comparing the two sorts and their performance. Be sure to comment on the relative numbers of exchanges and comparison in the various runs, the effect of the order of the data, the effect of different size files, the effect of different partition sizes and pivot selection methods for **Quicksort**, and the effect of using a **Natural Merge Sort**. Which factor has the most effect on the efficiency? Be sure to consider both time and space efficiency. Be sure to justify your data structures. Your analysis must include a table of the comparisons and exchanges observed and a graph of the asymptotic costs that you observed compared to the theoretical cost. Be sure to justify your choice of iteration versus recursion.Consider how your code would have differed if you had made the other choice.

It is to your advantage to turn the lab in on time. It will not be accepted after the due date except by prior arrangement.