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 Merge Sorts and a Heap Sort. They should both be recursive or both iterative, so that the overhead of recursion will not be a factor in your comparisons. Be sure to justify your choice. Also, in your analysis, consider how your code would have differed if you had made the other choice.

You will write three variations of merging to compare with Heap Sort: A two-way merge, a three way merge and a natural merge.

- A straight 2-way Merge, as discussed in the lecture, is the basis for the external sorting techniques, which become more relevant as the size of data increases. Two sorted sub files are combined to create a larger sorted file. Files of size one are trivially sorted, so you start with that. It requires extra space.
- A three-way merge combines three sorted sub files into one larger sorted file.
- A natural merge leverages partially sorted data. You may implement the natural merge with arrays or linked lists, but you need to justify your selection.
- Heap Sort is a practical sort to know and is based on the concept of a heap. It has two phases: Build the heap and then extract the elements in sorted order from the heap. Use the approach from the lecture.
- If time permits, consider a 4-way merge.

Create input files of four sizes: 50, 500, 1000, 2000 and 5000 integers. For each size file make three versions: randomly ordered data, reverse ordered data, and ordered data. Input files are available for your use or you can write your own. The provided files may require modification. To create the randomly ordered files, you may use a random number generator, but it is important to limit the duplicates to <1%. Alternatively, you may write a shuffle function to randomize an ordered file. It is to your advantage to add larger files or additional random files to the input - perhaps with 15-20% duplicates. You have an input set of 15 files plus whatever else you deem necessary and reasonable. **Each sort must be run against all the input files**. There are four sorts (Heap sort and three Merges) and 15 input files so **you will have a minimum of 60 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 files for all orders of size 50, and the output files for all inputs 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. At the end of each run, you need to print the number of comparisons and the number of exchanges but not the sorted data.

Turn in an analysis comparing the 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, and the effect of different merge types. Which factor has the most effect on the efficiency? Be sure to consider both time and space efficiency. 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.

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

Since the analysis of the asymptotic cost is the focus of this lab, **you may download code from other sources** for this assignment as long as **you give proper credit.**