CS 3130 Design and Analysis of Algorithms Fall 2015

Programming Project #2 [120 points].

Due date: Monday, October 26.

A. [30 points] Working with Merge Sort.

Suppose that an ordered file of size N is to be combined with an unsorted file of size M, with M much smaller than N. One approach is 'resorting': to append the small file to the original file, and sort a new file. Another approach is to sort a smaller file, and merge two sorted files. Determine in your experiments which method is faster, and how much faster.

Run your program for random arrays with $N=10^3, 10^4, 10^5$ elements and M=25, 50, 100. For each combination of N and M run your program 10 times and find average times.
In the script' file show that your function for Merge Sort works properly for a random array of 100 elements, and show ONCE how you sort a smaller file and merge it with the bigger sorted file. For all sorting procedures use the Merge Sort algorithm.

B. [25 points] Working with Quick Sort.

Implement 3 versions of a Quick Sort: (a) the basic version when a partitioning element is the rightmost element of the array; (b) the "median-of-three" Quick Sort; (c) find the median element from 5 randomly chosen elements of your array and swap it with the rightmost element, so that it will become a new partitioning element.

Run your program 10 times for each random array of the following sizes: $10^2, 10^3, 10^4$.

In the 'script' file show how each algorithm works for the file containing 50 random elements and display the sorting times of all three algorithms for one array of size 10^3.

C. [25 points] Working with Heap Sort.

Implement a heapsort for an array of N random integers. Build a heap using the 'bottom-up' method. Empirically determine the percentage of time that heapsort spends in the construction phase for $N=10^3, 10^4, 10^5, 10^6$. For each value of N run your program 10 times and find average times. In the 'script' file show how heapsort sorts the file of 50 random elements and display the 'construction' time and 'actual sorting' times for one array of size $N=10^3$.

Submit 'script' files for parts A, B, C through My Gateway.

Please, submit written analysis for all three parts:

Part A [15 points]. Submit theoretical analysis of efficiency of Merge Sort and results of your experiments. Theoretical analysis should include the recurrence for the Merge Sort and its solution. Experimental results should be presented in the form of tables and graphs with conclusions.

Part B [15 points] Submit theoretical analysis of efficiency of Quick Sort and results of your experiments. Theoretical analysis should include the recurrences for 3 different cases for Quick Sort and explanation why other versions of this algorithm may be helpful. Also explain how you select a partitioning element for versions (b) and (c) and how efficiency of the algorithm may be changed because of this choice. Experimental results should be presented in the form of tables and graphs with conclusions.

.

Part C [10 points]. Submit theoretical analysis of efficiency of a heapsort, including bottom-up and top-down heap construction, and your experimental results in the form of tables and graphs with conclusions.