CS 325 Spring 17
HW 1 – 25 points

1) (1 pt) Suppose we are comparing implementations of insertion sort and merge sort on the same machine. For inputs of size n, insertion sort runs in $8n^2$ steps, while merge sort runs in 64nlgn steps. For what values of n does insertion sort run faster than merge sort?

Note: lg n is log "base 2" of n or $\log_2 n$. There is a review of logarithm definitions on page 56. For most calculators you would use the change of base theorem to numerically calculate lgn.

2) (5 pts) For each of the following pairs of functions, either f(n) is O(g(n)), f(n) is Ω(g(n)), or f(n) = Θ(g(n)). Determine which relationship is correct and explain.

a. $f(n) = n^{0.25}$; $g(n) = n^{0.5}$
b. $f(n) = n$; $g(n) = \log^2 n$
c. $f(n) = \log n$; $g(n) = \ln n$
d. $f(n) = 1000n^2$; $g(n) = 0.0002n^2 – 1000n$
e. $f(n) = n\log n$; $g(n) = n\sqrt{n}$
f. $f(n) = e^n$; $g(n) = 3^n$
g. $f(n) = 2^n$; $g(n) = 2^{n+1}$
h. $f(n) = 2^n$; $g(n) = 2^{2^n}$
i. $f(n) = 2^n$; $g(n) = n!$
j. $f(n) = \lg n$; $g(n) = \sqrt{n}$

3) (5 pts)

a. Describe in words and give pseudocode for an efficient algorithm that determines the maximum and minimum values in a list of n numbers.
b. Show that your algorithm performs at most 1.5n comparisons.
c. Demonstrate the execution of the algorithm with the input L = [9, 3, 5, 10, 1, 7, 12].

4) (4 pts) Let f_1 and f_2 be asymptotically positive non-decreasing functions. Prove or disprove each of the following conjectures. To disprove give a counter example.

a. If $f_1(n) = O(g(n))$ and $f_2(n) = O(g(n))$ then $f_1(n) = \Theta(f_2(n))$.

b. If $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$ then $f_1(n) + f_2(n) = O(\max\{g_1(n), g_2(n)\})$

CS 325 Spring 17
HW 1 – 25 points

5) (10 pts) **Merge Sort vs Insertion Sort**

The goal of this problem is to compare the experimental running times of two sorting algorithms. These are very common algorithms and you may modify existing code if you reference it.

a) Implement merge sort and insertion sort to sort an array/vector of integers. To test the algorithms you will need to generate arrays of random integers. You may implement the algorithms in the language of your choice. Provide a copy of your code with your HW. Since we will not be executing the code for this assignment you are not required to use the flip server.

b) Use the system clock to record the running times of each algorithm for n = 1000, 2000, 5000, 10,000, …. You may need to modify the values of n if an algorithm runs too fast or too slow to collect the running time data. If you program in C your algorithm will run faster than if you use python. You will need at least seven values of t (time) greater than 0. If there is variability in the times between runs of the same algorithm you may want to take the average time of several runs for each value of n.

c) For each algorithm plot the running time data you collected on an individual graph with n on the x-axis and time on the y-axis. You may use Excel, Matlab, R or any other software. Also plot the data from both algorithms together on a combined graph. Which graphs represent the data best?

d) What type of curve best fits each data set? Again you can use Excel, Matlab, any software or a graphing calculator to calculate a regression equation. Give the equation of the curve that best "fits" the data and draw that curve on the graphs of created in part c).

e) How do your experimental running times compare to the theoretical running times of the algorithms? Remember, the experimental running times were "average case" since the input arrays contained random integers.

EXTRA CREDIT: *It was the best of times it was the worst of times…*

Generate best case and worst case input for both algorithms and repeat the analysis in parts b) to d) above. Discuss your results.