CISC621 Algorithm Design and Analysis, Fall 2015
Homework set 2, due Wednesday, October 14, 5:30pm

Check the "homework sheet" from the syllabus for general homework details. In particular, each homework solution is on an entirely separate (set of) sheet(s) of paper and is identified with your name(s). Do not staple solutions to two or more problems together. Submit in 201 Smith Hall directly to Chunbo Song or place on shelf marked CISC 621.

5. [Individual Problem] We saw that in divide and conquer sorting algorithms, such as merge sort and quick sort, it pays to shift to insertion sort when the array size, n, is small. For instance, by experiment on random arrays, when $n = 10$ insertion sort seems to pay off. Yet insertion sort has a worst case of 45 comparisons when $n = 10$,

   (a) whereas mergesort uses how many comparisons?

   (b) The theoretical minimum number of comparisons when $n = 10$ is 22, since $21 < \lg(10!) < 22$. Construct a sorting method that uses at most 23 comparisons when $n = 10$. Take care to clearly explain your method and it's correctness. Hint: it may help to find an optimal solution for 5.

6. [Group Problem] Consider the following game: I think of a positive integer: n, which you are to determine by guessing numbers and being told whether each guess is too high or too low or is n. There is no a priori upper bound on n – I am free to pick any n.

   (a) Give a strategy for determining n using at most $2\lg(n)+c$ guesses (c is a constant that you can choose).

   (b) But maybe I don't always tell the truth (or there is static on the communication line). Give a strategy for determining n using at most $2\lg(n) + 3lg(lg(n)) + k$ guesses (k is a constant you choose) on the assumption that at most one of the answers to your guesses will be a lie.

7. [Group Problem] Sometimes it is necessary to combine priority queues or break one into pieces. For instance on a multicore, it is an emerging practice to shut down some cores for energy efficiency when the load gets low. In this case you would merge the process priority queues of a shutting down core with that of a still active core Conversely, when load gets high, we want to split the priority queue of a busy core and give half it's process queue to a waking up core.

   (a) For the (min) binomial heap representation of priority queue, give efficient implementations of `merge` (combine two binomial heaps into one) and `split` (split a single binomial heap of size n into two heaps of size $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil$.

   (b) In some applications, there are many more insert operations than extract-min operations. Show that a series of n insert operations in a row, beginning with an empty binomial heap, costs $O(n)$