Due Tuesday, June 1.

Please read my policies concerning homework on the course website:
http://www.cs.duke.edu/courses/summer10/cps130/.

Recurrence Relations

Problem 1: Solve for the “big O” complexity of each of the following recurrence relations, using any method you choose. Show your work.

a) $T(n) = T(n/2) + n$

b) $T(n) = 7T(n/2) + n^3$

c) $T(n) = nT(n-1)$

d) $T(n) = T(n-1) + \log n$

Divide and Conquer

Problem 2: You are given array $A$ containing $n$ integers with no duplicates. The integers are sorted, but not necessarily sequential. You want to find out if there exists an index $i$ for which $A[i] = i$. For example, if $A = [-17, -3, 1, 4, 6, 20]$, then $A[4] = 4$ (assuming 1-based indexing). Give a divide-and-conquer algorithm that runs in time $O(\log n)$.

*Problem 3: A $k$-way merge operation. Suppose you have $k$ sorted arrays, each with $n$ elements, and you want to combine them into a single sorted array of $kn$ elements.

a) Here’s one strategy: Using the merge procedure from Mergesort, merge the first two arrays, then merge in the third, then merge in the fourth, and so on. What is the time complexity of this algorithm, in terms of $k$ and $n$?

b) Give a more efficient solution to this problem, using divide-and-conquer.

Problem 4: Suppose that at each run of Quicksort, the splits are in the proportion $1 - \alpha$ to $\alpha$ for some constant $0 < \alpha < 1/2$. Show that the minimum height of the recursion tree is approximately $-\frac{\log n}{\log \alpha}$, and that the maximum height is approximately $-\frac{\log n}{\log(1-\alpha)}$. 
Binary Search Trees

**Problem 5:** A binary tree is full if all of its nodes have either zero or two children. Let $B_n$ denote the number of full binary trees with $n$ nodes.

a) By drawing out all full binary trees with 3, 5, or 7 nodes, determine the exact values of $B_3$, $B_5$, and $B_7$.

b) Derive a recurrence relation for $B_n$.

c) Show by induction that $B_n$ is asymptotically lower bounded by $2^n$ (e.g., that $2^n = O(B_n)$, or equivalently, $B_n = \Omega(2^n)$).

**Problem 6:** Write (in pseudocode) recursive traversal functions that compute the following measures of a binary tree:

a) the size;

b) the height;

c) the path length.

Sorting

**Problem 7:** Suppose you must implement a sort algorithm in a language with no random access arrays, only linked lists. What algorithm would you choose to implement, and why? What changes (if any) would need to be made to the algorithm to accommodate the use of linked lists instead of arrays?

**Problem 8:** Implement Quicksort in the language of your choice. (Attach a code listing. If you collaborate with another student, only one of you need attach the code; just tell me who you collaborated with.) You can use whatever data structures you wish, although I recommend using the most basic array type your language offers. You should implement a random pivot, but for the problems below you will need to be able run with or without using a random pivot.

Now try your code out on:

a) a randomly generated sequence of $n$ integers, with and without the random pivot; and

b) some (2-3) non-random integer sequences, with and without the random pivot. For non-random integer sequences, I suggest taking some jpeg files and reading in the first $n$ bytes as integers. Another good source would be to take an already sorted or partially sorted word list - for instance, a recursive directory tree listing - and treat it as an integer array.

Be sure to make $n$ large enough to require a measurable amount of time for sorting. Time your calls to quicksort; you’ll get best results using timing code in your program or the unix `time` command, but a stopwatch will work, too. As an alternative, you might consider having your program print out the maximum recursion depth it reaches. Write up your results.

The point of this exercise to make the study of algorithms less abstract, and give you a feel for what really happens when the algorithm is run on “real” inputs. I think it is important to take algorithms into the real world to find out if their theoretical performance is borne out in practice. In no way is this intended as a test of your programming skill; I will not be grading for style or efficiency or whatnot. I strongly encourage you to try coding this yourself, but to ask for help if you get stuck - don’t spend a long time programming and debugging.