Recurrence Relations

Problem 1: (20 points) 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: (20 points) 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: (30 points) 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: (10 points) 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: (10 points) For the set of keys \{1, 4, 5, 10, 16, 17, 21\}, draw binary search trees of height 2, 3, 4, and 6.

Problem 6: (20 points) Prove that inserting a single item into an AVL tree requires no more than 2 rotations to maintain the AVL tree balance property:

(a) Show that a single insertion results in a balance factor no worse than ±2 at any node, and that all such out-of-balance nodes must lie on the path from the inserted node to the root.

(b) Show that two or fewer rotations are required to repair the balance at the deepest out-of-balance node \(\nu\), and that the result of this operation reduces the height of the tree rooted at \(\nu\) by 1.

(c) Use (a) and (b) to show that rebalancing the deepest out-of-balance node rebalances the entire tree.

Sorting

*Problem 7: (20 points) 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 non-random integer sequences, with and without the random pivot. Try an already sorted list, and then try some unsorted but non-random input, such as the first \(n\) bytes of a large text document (e.g., from http://www.gutenberg.org/).

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 \texttt{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.