Week 13, Lecture 2

Today we'll only look at four program files and thus finish the last topic in this course, which is recursion. The last program (program 4, Quicksort) we'll do is something extra, and you are not required to know it, though it is similar to the MergeSort (program 3) which is in your book. You should know the insertion, selection, and merge sorts; and you should know the linear and binary searches. And you should know how the recursive versions for some of these , i.e., the ones we have covered (excluding Quicksort).

Program 1 shows you a simple recursive method to compute x^n (some number x raised to the power of an integer n). Instead of multiplying x out (n-1) times (which would be an O(n) computation), we can use "divide and conquer". Think of the "tree" structure and visualize x^n as the root of the tree. Keep it simple at the start and assume n is even.

Then the left child of the root is x^(n/2) and the right child of the root is x^(n/2).

The idea is that you compute x^(n/2) just once. Then multiply it by itself to get x^n.

But we still have to compute x^(n/2). Assume for a moment that n/2 is also even. Then the same procedure we used above should work. In fact this procedure should work until we get to the bottom of the tree. In other words, the whole task is done in O(log n) steps (where the log is to the base 2). Do you see how the pattern has so much in common with the binary-search and merge-sort structures? They all follow the same “tree” pattern during computation. Trees and recursions are good friends.

Whenever we arrive at a case where n or n/2 or n/4 etc. is not even, we follow exactly the same procedure as before but do one extra multiplication. Because, for example, x^5 = x^2 * x^2 * x, where the last multiplication (i.e., “ * x ”) is the extra multiplication to handle the “odd” value.

This is useful when you raise a matrix to some large power, because each matrix multiplication is an O(n^3) operation.

Program 2 is a simple example of how you can reverse a string recursively.

Program 3 is the Merge-sort (again). We go through the same example as before, but step by step, so that you see exactly what happens.

Program 4 is the (famous) Quicksort. You are not required to know it, but you’ll see that it is simple, and looks a lot like the merge sort, except without all the list copying.

NOTE: We will not have class during ThanksGiving week.

If I have some time I may post some additional, off-course material (perhaps a small video) next week. But if not, I will see you the week after ThanksGiving — where we will cover material which you will not need for your exam. This last bit will mainly cover classes and objects. You will see how it makes your code much cleaner, besides having many other advantages.