Computer Science 308-250B Homework #4

Due Monday March 24, 2003, 17:00

[20 %] Question 1. [Recursive matrix multiplication revisited] Which of naive recursive matrix multiplication (as in Question 3 of the midterm) or Strassen’s algorithm for matrix multiplication (as in the slides of week 3, lecture 2) is more efficient, in terms of asymptotic complexity (big-O, etc.)? Provide a proof.

[30 %] Question 2. [Graph ADT] Let \( G = (V, E) \) be a connected graph. A “spanning tree” \( T = (V', E') \) is a tree such that

\[
\begin{align*}
V' &= V \\
E' &\subseteq E
\end{align*}
\]

and such that every vertex in \( V \) that is part of an edge in \( E \) is also part of an edge in \( E' \).

Write a program that, given a graph \( G \) in adjacency list representation, outputs a spanning tree for this graph in \( O(m) \) time, where \( m \) is the number of edges in \( G \). Informally explain why this is impossible to do in \( O(m) \) time if we were using adjacency matrices.

[50 %] Question 3. [Polynomial ADT] In this question we will implement the basic representation of polynomials in one variable with non-negative integer exponents and real coefficients. You already have access to the pseudocode for all the the operations defined for this ADT in the slides of Lecture 7-3, except for multiplication, which you must do from scratch. Also, improve the pseudocode for \textbf{Evaluate} so that it runs in time at most \( 2d \) where \( d \) is the degree of the polynomial, and provide a proof of this property (note that you should count only the total number of additions and multiplications). Write your implementation inside the provided class skeleton \textbf{Poly}.

In this assignment I have given you skeleton for a main method, \texttt{polyTest}, which I want you to run to test the program. You must first complete two methods in this testclass: \texttt{parsePoly} and \texttt{printPoly}. In this case I want you to capture the output to a file called \texttt{foo}. In order to do this you do the following:
java polyTest > foo

Your diskette (or WebCT submitted files) should contain the .java file and all .class files as well as the file called foo. Please do not edit foo with an editor even to clean it up. Any attempt to forge the output with an editor will be treated as attempted fraud and will result in disciplinary action.

[5 %] **Bonus A.** Provide the pseudocode for and implement the remaining operation, getRoot.

[5 %] **Bonus B.** Provide the pseudocode for and implement an additional operation, Factorize. Note that this is equivalent to finding the exact roots. You may assume that the coefficients are integers (note that this is equivalent to having rational coefficients). You also may assume that we are only interested in integer (or rational) roots.