CS383, Algorithms, Spring 2012, Midterm

This is an open book exam. No computers are allowed. You have 1 hour and 15 minutes to complete the following 5 questions. If a question does not ask for pseudocode, you may use plain English to describe your algorithm. Try to make your answer compact and to the point. Please write your answers on the exam book. Partial credits will be given, so try to write down your solutions even when you are not sure.

1. In insertion sort, we follow such a procedure: we first find the smallest element in array \( A \) and exchange it with \( A[1] \), then we find the second smallest element and exchange it with \( A[2] \). We repeat the procedure \((n - 1)\) times where \( n \) is the number of elements in \( A \). Write pseudocode for this algorithm. Analyze its worse case complexity.

2. To compute a polynomial \( P(x) = \sum_{k=0}^{n} a_k x^k \), we can use the following method:

\[
P(x) = a_0 + x(a_1 + x(a_2 + ... + x(a_{n-1} + xa_n)))
\]

The algorithm’s pseudocode is as follows:

\[
y := 0
\]

\[
i := n
\]

\[
\text{while } i \geq 0 \text{ do}
\]

\[
y = a_i + x \cdot y
\]

\[
i = i - 1
\]

\[
\text{end while}
\]

(a) Show that this code is correct.

(b) What’s the asymptotic running time of this code, assuming that your machine can do addition and multiplication in a single step?

(c) How does it compare with directly evaluating the polynomial?

3. A binary tree is a rooted tree in which each node has at most two children. Using induction to prove that in any binary tree the number of nodes with two children is exactly one less than the number of leaves.

4. Let \( M \) be an \( n \times m \) matrix. \( M \) has been sorted in each row and each column in the ascending order: Each element is not less the one on the left and the one above. Design an efficient algorithm to find whether a number is in the matrix. If it is not in the matrix, your algorithm should report it. Give the complexity of your algorithm in big \( O \) notation.

5. An arborescence of a directed graph is a rooted tree so that there is a directed path from the root node to each node of the graph. Give an efficient algorithm to test whether a graph has a arborescence. Give the complexity of your algorithm.