Data Structures and Algorithms

Problem Sheet 1

1. List the functions below from lowest order to highest order. If any two (or more) are of the same order, indicate which.

n	$n - n^3 + 7n^5$	$n^2 + \lg n$	n^3
2^n	$\lg n$	n^2	$(\lg n)^2$
$n \lg n$	$\sqrt{n}$	2^{n-1}	$n!$
$\ln n$	e^n	$\lg \lg n$	$n^{1+\epsilon}$

2. Prove or disprove: For any positive constant c, $f(cn)$ is $\theta(f(n))$.

3. Prove or disprove: $\sum_{i=1}^{n} i^2 = \theta(n^2)$.

4. Prove the following. There are at most 2^ℓ nodes at level ℓ of a binary tree. A binary tree with depth d has at most $2^{d+1} - 1$ nodes. A binary tree with n nodes has depth at least $\lfloor \lg n \rfloor$.

5. Consider the following variant of Insertion Sort: for $2 \le i \le n$, to insert the key $L[i]$ among $L[1] \le L[2] \le \cdots \le L[i-1]$, do a binary search to find the correct position for $L[i]$.
   - How many key comparisons would be done in the worst case? (using big-Oh.)
   - What is the total number of times keys are moved in the worst case? (using big-Oh)
   - What is the order of the worst case running time?

6. The first n cells of an array L of size N (much bigger than n) contain integers sorted in increasing order. The remaining cells all contain some very large integer that we may think of as ∞. The array may be arbitrarily large, and *you don't know* n. Give an algorithm to find the position of a given integer $x < \infty$ in the array in $O(\lg n)$ time.

7. We observed that a worst case for Insertion Sort occurs when the keys are initially in decreasing order. Describe some other initial arrangements of the keys that are also worst cases.

8. Solve the following recurrences. Assume that $T(1) = 1$.

   (a) $T(n) = T(n/2) + c \lg n$.

   (b) $T(n) = \sqrt{n}T(\sqrt{n}) + n$.

   (c) $T(n) = T(n/2) + cn$.

   (d) $T(n) = 2T(n/2) + cn$.

   (e) $T(n) = 2T(n/2) + cn \lg n$.

   (f) $T(n) = 2T(n/2) + cn^2$.

9. $T(n) = T(n/3) + T(2n/3) + n$; $T(1) = 1$. Show that $T(n)$ is $\Omega(n \lg n)$ using the recursion tree.

10. M is an $n \times n$ integer matrix in which the entries of each row are in increasing order (reading from left to right) and the entries in each column are in increasing order (reading top to bottom). Give an efficient algorithm to find the position of an integer x in M, or determine that x is not there. Say how many (3-way) comparisons of x with matrix entries your algorithm does in the worst case. Is your method optimal?

11. M is an $n \times m$ integer matrix. Suppose you sort each row of M in ascending order, and then sort each column of M in ascending order. Show that after this the rows remain sorted.