# PhD Qualifier Exam for Fall 2007— Theory Area

CS Department, IIT

**Your number**: _____

There are 4 questions in this exam. For every question, please write your answer in a clean and concise way.

If you are asked to write an algorithm for a question, you have to write the **pseudo-code** of your algorithm and also put explanations about your pseudo-code. Also show correctness and estimate the running time. Every statement must be proven (or be obvious to the grader).

1. A subsequence is a palindrome if it is the same when read left to right and right to left. A subsequence does not have to be contiguos. Describe a polynomial-time algorithm to find the longest subsequence which is a palindrome in a given string represented by an array $A[1..n]$.

   For example, the string *abcab* has four palindromes of length 3: *aba*, *aca*, *bcb*, and *bab*, but no palindrome of length 4.

2. Describe a binary search tree on $n$ nodes such that the average depth of a node in the tree is $\Theta(\lg n)$ but the height of the tree is not $O(\lg n)$. How large can the height of an $n$-node binary search tree be if the average depth of a node is $\Theta(\lg n)$?

3. In an adjacency list representation of an undirected graph, it is useful to have a pointer from each of the two entries for an edge to the other. Provide a linear-time (i.e., $O(m+n)$) linear-space algorithm to create such an adjacency list representation from a standard adjacency list representation.

4. Assume that you *only* know the following problems are NP-complete: SAT, 3SAT, VERTEX-COVER, CLIQUE, HAM-CYCLE, SUBSET-SUM (for the definition of the problems, look at the included chapter of Cormen et. al)

   Consider the following problem, called DOMINATING SET: Given a graph $G = (V, E)$, and an integer $K$, is there a set of vertices $A \subseteq V$ such that every vertex of $V$ is either in $A$ or has a neighbour in $A$.

   Prove that the DOMINATING SET problem is NP-complete. **Hints**: If you plan to use SAT, use three vertices for each variable and one vertex for each clause. If you plan to use VERTEX-COVER, add $m + 2$ vertices, where $m$ is the number of edges in the original instance.

   Ofcourse, there are correct solutions which ignore the hints.