CS 466/666
Assignment 1
Due: Noon Thursday Sept. 25, 2008

1) [10 marks] Prove that the dynamic programming algorithm given in class, to find the optimal binary search tree for a given sequence of probabilities, runs in $\Theta(n^2)$ time. The key issue is in restricting the subrange in which we search for the root of an optimal tree for a given range. You may assume the lemma that the optimal subtree on the range i to j cannot have a root to the right of the optimal on i to j-1. For clarity of your proof, first state the algorithm in pseudo code.

2) [10 marks] Recall the discussion of finding a "near optimal" binary search tree, in linear time. The point of this exercise is to complete a proof of the linearity, i.e. that the number of comparisons (in the worst case) is of the form $cn \pm o(n)$. To make things a bit cleaner, let us assume we are given a set of values and their probabilities of access, so the probability of falling between two consecutive values is 0. You are to determine the number of comparisons required by this method, in the worst case, as accurately as possible. The method was claimed to take $\Theta(n)$ time, determine the constant (c above) , and, to the extent you can, also determine any lower order term (the o(n) above). Prove your claims.

3) [5 marks] Consider the context free language and the grammar
S ::= AB | BC A ::= BA | a B ::= CC | b C ::= AB | a
Use the dynamic programming method discussed in class to show the string baaba is in this language.

4) [10 marks] Suppose we want the virtues of both an optimal binary search tree and a balanced search tree. Give a method of producing a tree with cost "close" to optimal but with no search costing more than $\Theta(\lg n)$. Prove your tree achieves these claims.

5) [15 marks] Suppose you are given an acyclic directed graph (with n nodes and m edges), in which each edge has a distinct weight (say the weights are the integers in the range 1, …, m). Give an efficient algorithm to find the longest path in the graph with increasing edge weights. (If there is a tie any of the longest paths will do). A dynamic programming "point of view" may help. What is the runtime of your method? Justify this claim.
In the example below, there is a path of length 3 (a→b→d→e) but the edge weights (6,4,3) are not increasing. The longest path with increasing edge weights is a→d→e.