CS 466/666
Assignment 1
Due: Noon Friday October 1, 2010

1) [32 marks] We have discussed the relationship between multiplication of an adjacency matrix and graph properties. This question explores the relation somewhat further. Throughout this question assume:

a. A is the adjacency matrix of the n node (directed) graph G. $\underline{B}$ denotes the matrix B in which all non-zeroes are replaced by 1's.
b. "Words" are of length lg n bits. Multiplying two single precision words takes 1 operation; multiplying two k precision (i.e. k lg n bit) integers takes k lg k operations (the fastest known method is just a little worse).
c. n by n matrix multiplication takes M(n) arithmetic operations each of the precision of the largest value of the given matrices. (Single precision operations if the given matrices are (0,1)).

In each part of this question give justification for all answers, including runtimes.

i. [4 marks] What does the [i,j] entry of A^2 indicate about graph G? What does $\underline{(A^2)}$ indicate about G? (Square A then replace all non-zeroes with 1's)
ii. [4 marks] What does the [i,j] entry of A^n indicate about graph G? What does $\underline{(A^n)}$ indicate about G?
iii. [8 marks] How many bits are required to represent the largest possible value in A^n? How would you ("efficiently") compute A^n and how many single precision operations are required? An answer up to Θ notation suffices.
iv. [8 marks] How would you, more efficiently than from A^n, compute $\underline{(A^n)}$? How many single precision operations would this require?
v. [4 marks] Justify the claim that if A has a Hamiltonian cycle, all entries on the diagonal of $\underline{(A^n)}$ are 1.
vi. [4 marks] Do the last two parts suggest a polynomial time algorithm for the Hamiltonian cycle problem? Why or why not?

2) [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.

3) [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.

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.