CS 466/666
Assignment 5
Due: Noon Monday December 6, 2010

1) [5 marks] Give a linear time algorithm that finds an optimal vertex cover for the special case that the graph is a tree.

2) [8 marks] Consider the following greedy algorithm for finding a vertex cover on a graph: Sort the vertices of the graph in non-increasing order according to the degrees. Iteratively include the next vertex on the list to the vertex cover and verify if the set is indeed a vertex cover. If so, declare the set as a vertex cover, otherwise continue with the next vertex on the list (note that unlike the 2-approximation discussed in class, no edge is deleted in the process). Show that the algorithm does not necessarily approximate the optimal vertex cover by **ANY** constant factor; actually, find an example making it as bad as you can.

3) [6 marks] Suppose that the vertices of an instance of the traveling-salesman problem are points in the plane and the cost of an edge is the Euclidean distance between its endpoints. It was claimed in class that an optimal tour never crosses itself. Prove this.

4) [15 marks] In discussing the move to front heuristic, we observed that if the distribution of requests is very skewed (such as $p_i = 1/2^i$) it may lead to faster search times than (on an expected or even an amortized basis) than keeping the list in sorted order by key and doing a binary search. For purposes of this problem, we will assume that element comparisons are the only cost incurred in doing a search, and that a binary search takes 1 + lg n probes. Furthermore, we will think of n as being extremely large.

Suppose we are to do a search on a set where the probability of a request for the i^{th} most frequent value is proportional to i^{-2}; hence $p_i = 1/(i^2 H_2(n))$, where $H_2(n) = \sum_{i=1..n} 1/i^2 \approx \pi^2/6$ for large n. So you can take p_i as $6/(i^2\pi^2)$. This distribution is sometimes called Lotka's law.

a) Find the expected cost of a linear search for an element from this distribution if values are stored in decreasing order by probability. Get the correct constant factor of the lead term in your answer (i.e. if the answer is $7\sqrt{n} + \lg n$, then the answer $7\sqrt{n}$ is satisfactory, but $\Theta(\sqrt{n})$ is not. Clearly a little calculus will go a long way)

b) It turns out that in this case the expected cost of MtF is $\pi/2$ times the static optimal (i.e arranging the elements in decreasing order by probability of being accessed.). We proved this ratio is no worse than a factor of 2. But if queries come from a fixed independent distribution it turns out that the distribution given above is the worst case. With that verbose preamble, what is the expected cost of the MtF for this (Lotka's) distribution, and how does it compare with binary search? Pay attention to the constant in front of the lead term.

c) Given that the probabilities above are indicative of the frequencies over a long sequence of accesses, under what circumstances (i.e. in what order are requests made, is the application a real time application or a batch system) would binary search be far superior? Under what circumstances would MtF be far superior?