# CS 466/666 Spring 2013
# Assignment 3
# Due Noon, June 3, 2013

You are on your honour to present your own work and acknowledge your sources.

1. [8 marks: Using Doubling Binary Search] This question finishes up the proof that the approximation to the optimal binary search tree can be found in linear time if the probabilities of access to keys (and for unsuccessful searches for values between consecutive key values) are given in order by the key values. Having made this introduction we pose the problem in a straightforward mathematical form.
   a. [3 marks] State the recurrence for the time, $T(n)$, for the algorithm outlined below:
      -You take time lg $i$ to break a problem of size $n$ into 2 subproblems of sizes $i-1$ and $n-i$.
      -You have no control over $i$, other than it is an integer in the range $[1,n/2]$.
      -Problems of size 0 and 1 take no time.
   b. [5 marks] Show that the solution to the recurrence (and so runtime of the algorithm) above is $O(n)$. (Note: If you plug in $T(n) = cn$ to the recurrence expected, it will not lead to the correct solution. Remember that proving things by induction often requires strengthening the induction hypothesis. In this case you will want "$T(n) = cn$ – something", for specific values of "c" and "something".)

2. [4 marks: Near-optimal Binary Search Tree] This question also deals with the "approximately optimal" binary search tree heuristic. The method we discussed works as follows.
   -Choose the root by a greedy heuristic and recursively solve the problem for the left and right subtrees.
   - The root chosen has the key value such that the maximum of the probability of being in its left subtree and the probability of being in its right subtree is minimized.

   Show by giving an explicit counterexample (i.e. the $p_i$'s and $q_i$'s), that the tree so formed is not <u>necessarily</u> the optimal binary search tree for the probabilities given.