CS 466/666
Assignment 4
Fall 2005
Due date: November 25th, 6:00pm

1. Consider an n x n array A storing n^2 elements such that the elements in each row appear in nondecreasing order from left to right and the elements in each column appear in nondecreasing order from top to bottom. The goal is to determine where a given element x can be found in A or to determine that it is not present. In the questions below, comparisons refer to comparisons between x and elements in A, where the result of the comparison can be $<$, $=$, or $>$ for $x < A[i,j]$, $x = A[i,j]$, and $x > A[i,j]$, respectively. Comparisons between pairs of array elements are not allowed.

   a. Give an algorithm that solves the problem; determine the exact (not asymptotic) number of comparisons needed in the worst case. (*Hint: your answer should be linear in n).*
   b. Give a lower bound on the number of comparisons needed by any comparison-based algorithm that solves the problem.The lower bound should match your upper bound. *(Hint: your adversary should start with initial assignments of values to each entry, making minor modifications if necessitated by the choice of comparisons by the algorithm.)*

2. Consider the problem of binpacking in which you are given n objects $x_1, x_2, \ldots, x_n$ with associated weights $w_i \leq 1$. The goal is to package all items using as few bins as possible, where each bin has capacity 1.

   a. A simple algorithm is as follows: start with n empty bins $b_1, b_2, \ldots, b_n$, and process the items in the order given. Each item is placed in the first bin that has enough space left to include it. The solution consists of the non-empty bins at the end.
   Prove that this algorithm requires no more than twice as many bins as the optimum solution.
   b. A modification of the above algorithm is to first sort the items by decreasing size, and then process them as above, but in this sorted order.
   Prove that this algorithm requires at most 3/2 as many bins as the optimum (give or take one extra bin).
   c. Consider the algorithm which processes the n inputs in order, placing an item in the non-empty bin with the most unused space, if it fits, or on an empty bin otherwise.
   Give an upper bound on the approximation ratio for roomy fit in the special case in which for all i, $w_i \leq 1/2$.

3. In the KNAPSACK problem we are given n objects $x_1, x_2, \ldots, x_n$ with associated weights w_i and values v_i, plus a knapsack capacity W. The goal is to select a subset of the objects whose sum of weights adds to at most W, and whose sum of values is maximized.

   a. Construct examples to show that a greedy heuristic which always selects the item of highest value, if it fits, is not guaranteed to come within a constant factor c of the optimal choice, for any constant c.
   b. Consider a second heuristic which sorts the item by decreasing value per weight v_i/w_i. Again construct examples to show that this cannot come within a constant factor of the optimal choice.
   c. Show that if we run both the greedy heuristic from (a) and the heuristic from (b) independently and return the solution with highest value, then this is within a factor of 2 of optimum.

4. Consider the following algorithm for finding the median. Given a set of n elements $A[1], A[2], \ldots, A[n]$ select $i \in \{1,2,\ldots,n\}$ at random, then partition the list in those elements that are larger than $A[i]$ and those that are smaller than $A[i]$. Recurse on the larger set. Compute the expected time to find the median. *(Hint: establish a recursion for the expected time T(n) and then compare to T(n-1)).*

5. In order to ensure a chair and a secretary for each committee at the university, we wish to solve the following problem. We are given the rosters of k committees, where no two committees consist of the same subset of the n professors. Each committee has exactly p members, where $k \leq 2^{p-2}$. The goal is to assign to each professor the title "chair" or "secretary" so that each committee has at least one member with the title "chair" and at least one with the title "secretary". You may assume that there is always such an assignment.

Give a Las Vegas algorithm that solves the problem.

6. Given problem P, suppose you have a Monte Carlo algorithm A and a procedure that can verify the correctness of a solution in time $t(n)$. Moreover, suppose that the running time of A is at most $T(n)$ on any instance of size n and that A produces a correct solution with probability $\gamma(n)$. Show how to obtain a Las Vegas algorithm that runs in expected time at most $(T(n) + t(n))/\gamma(n)$.

7. A mountain patrol realizes that a climber has failed to return to base camp and it is presumed to be injured at some point in the trail. They contact a ski lodge midway along the trail and request that a search take place for the climber. The ski patrol does not know if the climber became injured before or after passing the lodge on the trail. To complicate matters further a snow storm has reduced visibility to nearly zero. Give an algorithm that is guaranteed to find the climber within a factor of 9 of the straight path to the climber, had we known on which side of the trail the climber was. You may assume that the climber is at least 1 meter away from the lodge.

8. An opposition party is trying to form the new government of the country. In the summer its technical IT staff wrote a program $Q(T)$ that, given an allotted amount of CPU time T, can compute the best possible electoral strategy using that much computational power (i.e. any better electoral strategy, if any, requires more CPU time to be found).

Originally the plan was to run this program with T equals "from now until the preannounced election time by the prime minister". However given current events, the programmers are now no longer sure when exactly their leader will make the government fall and an election will take place. Using your answer to question 7, propose a new algorithm R which uses $Q(T)$ as a subroutine to compute a reasonable electoral strategy regardless of the time in which it is called. The algorithm R runs until interrupted at which time it returns an electoral strategy.

State the quality of your solution in terms of the competitive ratio as compared to $Q(N)$, where N is the time at which the election eventually took place.