SCI 199 (Beautiful Algorithms)

Assignment #1 Due: Oct. 4, 2001 (in tutorial)

1. Consider the following 2-dimensional generalization of the bin packing (or cutting stock) problem that we saw in class. Instead of trying to use as few I-beams as possible to meet the set of orders, we now try to use as few large squares as possible to meet the set of orders where each order is now considered to be a square. More formally, we want to use as few $N * N$ squares as possible to fill the set of orders $\{n_1, n_2, \cdots, n_k\}$. To fill an order, squares of various sizes must be cut from some $N * N$ square. For example, if $N = 10$ and $n_1 = 8, n_2 = 6, n_3 = 2, n_4 = 2, n_5 = 3$, then 2 squares are needed. The first could contain n_1, n_3 and n_4 whereas the second square could contain n_2 and n_5.

   (a) Design a heuristic algorithm to solve this problem. Note that this problem is NP-complete, so your algorithm is not expected to use the minimum number of squares. Your algorithm will have to consider not only how to assign little squares to a big square but also how to organize the little squares inside the big one. Be careful to describe your algorithm carefully - it is not the tutor's job to try to debug your algorithm.

   (b) Construct an example to show that your algorithm is not optimal.

2. Here is a non-recursive algorithm to solve the HANOI problem on n disks. Place the polls, S, T and U in a circle with S at 12 o'clock. If n is odd, then place T at 4 o'clock and U at 8 o'clock. If n is even, then place U at 4 o'clock and T at 8 o'clock. Recall that the n disks start at S.

   (a) If disk 1 was not the last disk moved, then move disk 1 one position clockwise.

   (b) If disk 1 was the last disk moved, then make the only possible legal move that doesn't involve disk 1.

   Give a convincing argument that this algorithm will produce the same moves as the recursive algorithm presented in class.

3. EXTRA Modify the 3 pole algorithm presented in class to handle the case where there are 4 poles. (i.e. there are now 2 "utility" poles.) (Hint: to get a feeling for the algorithm, first consider how you would solve the problem when there are only 3 disks - then consider how you would solve it using 4 disks.)

   Is your algorithm optimal? (i.e. will it produce the minimum number of moves?) If so, give a convincing argument; if not, give a counter-example.