15-854 Approximation and Online Algorithms

Homework # 6 **due: April 24, 2000**

Note: This will be the last hwk before the take-home final.

Problems:

1. Finding your keys, revisited. We are looking for our keys in a graph G, starting from some start node s. We have seen that different performance measures lead to different classic algorithm problems. For instance, if we measure a strategy by the worst-case time to find our keys, then the optimal solution is the traveling salesman path. If we measure performance by the average-case time, under the assumption that the keys are at a random location (each node equally likely) then we get the traveling deliveryman problem (also called the minimum latency problem). If we measure performance by the worst-case ratio of our time to the actual length of the shortest path, then we get the setting in hwk5.

   For this problem, we consider the deliveryman problem (keys are at a random location and the cost of a solution is the expected time to find them) on an unweighted tree (all edges have length 1).

   (a) Prove that depth-first search has the same cost no matter how it makes its arbitrary choices. For instance, on a binary tree, at each node, DFS can choose to first go left or first go right. Prove that even if the tree is unbalanced, it doesn't make a difference which choice you make. (If you like, you can consider just binary trees rooted at the start node.)

   (b) Prove that, in fact, DFS is optimal for unweighted trees: it yields the optimum traveling deliveryman path. Hint: if we call the start node the "root", prove that for *any* strategy, the ith distinct vertex reached v will be reached at time at least $2i - depth(v)$.

2. (Due to E. Lauer). Consider the following game. A friend picks a secret number i by flipping a coin until it comes up heads (i.e., $i = 1$ with probability $1/2$, $i = 2$ with probability $1/4$, etc.) He then takes two slips of paper, and writes down 3^{i-1} on one piece and 3^i on the other. He then randomly puts one piece of paper in one hand and the other in the other hand. You get to pick a hand and see the number in it. Let's say you see the number x, and the other hand has the (unknown) number y. You now get to say either "keep" or "switch". The second hand is now opened: if you said "keep", then you win $x - y$ dollars, but if you said "switch", you win $y - x$ dollars.

   (a) Clearly if $x = 1$ then you should say switch. What if you see $x = 3$; what strategy maximizes your expected gain? What about general x?

   (b) The strategy "switch no matter what x is" is equivalent to the strategy "keep no matter what x is" since the pieces of paper were placed into the hands randomly. How do you reconcile this with your answer to part a? Try to give as satisfying an explanation as you can. Here are some interesting random variables to consider. Define X_i to be the probability that you are initially shown the number 3^i, times the expected gain of your strategy given that you are initially shown 3^i. Define

Y_i to be the probability that the pair of numbers chosen is $\{3^i, 3^{i+1}\}$, times the expected gain of your strategy given that this is the pair chosen.

(c) Say we have a ulility function $f(z)$ that assigns a value to winning z dollars (z might be negative), and our goal is to maximize expected utility. Can you redesign the game so that paradox still occurs for utility function $f(z) = sign(z)\lg(|z| + 1)$? [i.e., winning 99 dollars has utility $\lg(100)$ and losing 99 dollars has utility $-\lg(100)$].

(d) Can you define a utility function that is monotone increasing (it is always better to have more money) such that the paradox can no longer happen? (In particular, connect this to your explanation in part b).