CS 561, HW3

Prof. Jared Saia, University of New Mexico

Due: October 9th

1. Problem 4-5 (VLSI chip testing) - This is a really good divide and conquer problem I left out of the last hw

2. (h-trees) A h-tree is a rooted binary tree that is a useful data structure for designing self-healing networks (because they can be merged quickly). Let ℓ be a positive integer. For ℓ a power of 2, the complete tree with ℓ leaf nodes is the unique h-tree with ℓ leaf nodes. For ℓ not a power of 2, a tree with ℓ leaf nodes is a h-tree if and only if (1) the root node, r, has two children; (2) the left subtree of r is the root of a complete binary containing $2^{\lfloor \log \ell \rfloor}$ leaf nodes; and (3) the right subtree of r is a h-tree. Recall that a *complete* binary tree is one where every internal node has two children and every leaf node has the same depth.

   Show the following by induction:

   - For all positive ℓ, there is a unique h-tree with ℓ leaf nodes.
   - Call the h-tree with ℓ leaf nodes h-tree(ℓ). Then, the height of h-tree(ℓ) is $\lceil \log \ell \rceil$

3. Find the optimal parenthesization for a matrix-chain product whose sequence of dimensions is: $(2, 3, 2, 2, 2, 1)$. (Don't forget to include the table used to compute your result)

4. Show via induction that a full parenthesization of an n element expression has exactly $n - 1$ pairs of parenthesis.

5. A bakery sells donuts in boxes of three different quantities, x_1, x_2, and x_3. In the Donut Buying problem, you are given the numbers x_1, x_2 and x_3, and an integer n and you should return either 1) the minimum number of boxes needed to obtain exactly n donuts if this is possible,

along with a set of boxes that obtains this minimum; or 2) "DOH!" if it is not possible to obtain exactly n donuts.

For example if $x_1 = 4$, $x_2 = 6$, $x_3 = 9$ and $n = 17$, then you should return that 3 boxes suffices, with 2 boxes of size 4, and 1 box of size 9. However, if $n = 11$, you should return DOH! since it is not possible to buy exactly 11 donuts with these box sizes.

(a) For any positive x, let $m(x)$ be the minimum number of boxes needed to buy x donuts if this is possible, or INFINITY otherwise. Write a recurrence relation for the value of $m(x)$. Don't forget the base case(s)!

(b) Give an efficient algorithm for solving Donut Buying. How does its running time depend on x_1, x_2, x_3, and n? Is it an algorithm that runs in polynomial time in the input sizes?

6. Problem 15-5 (2nd)/ 15-7 (3rd) (Viterbi Algorithm). Note in this problem, a label can appear on more than one edge in the graph, and can even appear on more than one edge leaving a given node in the graph.

7. You are competing in the popular game show "Let's Make a Dynamic Program" with another player. You and your opponent both start with 0 dollars. If you reach (or exceed) n dollars before your opponent, you win n dollars; if your opponent reaches (or exceeds) n dollars before you, you win nothing; and if you both reach (or exceed) n dollars at the same time, you both win n dollars. In each turn, you get to choose the level of difficulty of the next question asked, where this difficulty is represented by an integer k between 1 and n. If you answer the question correctly, you get k dollars, otherwise your opponent gets k dollars. Note that you are always in control throughout the entire game of the difficulty level of the question asked.

   Through careful study of the game you have been able to determine for all i between 1 and n, the probability p_i that you will answer a question of difficulty i correctly.

   (a) Consider the greedy algorithm where you always choose a question of difficulty level i for i maximizing $i * p_i - i * (1 - p_i) = i(2p_i - 1)$. Is this an algorithm that is optimal in the sense that it maximizes your expected winnings? Hint: Is it ever better to make a long shot bet because the probability of success from multiple short bets is small. In particular, think about the case where your opponent has $n-1$ dollars and you have 0.

(b) Let state (i, j) be the state where you have i dollars and your opponent has j dollars. Note that if you choose the difficulty level to be k at that state, you have probability p_k of going to state $(i+k, j)$ and probability $(1-p_k)$ of going to state $(i, k+j)$. Now let $e(i, j)$ be your expected winnings if you have i dollars and your opponent has j dollars and you play optimally. Write a recurrence relation for the value $e(i, j)$. Note: You will find it useful to consider i and j values that range from 0 to $2n-1$. Hint: Use expected values for simpler subproblems and the probabilities p_i described above to compute $e(i, j)$. Don't forget the base case(s).

(c) Give the pseudocode for computing the value $e(0, 0)$, which gives you your expected winnings if you play this game optimally.

(d) *HARD:* What if the game is changed as follows. You still select the difficulty level k, but after your selection, both you and your opponent have the chance to write down the answer to the question. Whoever gets the answer correct wins k dollars (note that both of you may win now). There is no penalty for a wrong answer. The probability that you answer a question of difficulty k correctly is p_k and the probability that your opponent answers correctly is q_k. Can you still solve this new problem using dynamic programming? If so, give a recurrence and describe how to change the algorithm. If not, describe why not.