HW 12 textbook problems and hints

Note on author's hints: A student suggested that I put all of the **author's hints** at the end, so that students have a chance to think about the problems before they see the hints. I have done that for this assignment.

The original problems 1-2 were removed, and the later problems were renumbered on January 31, 2017

Problem 1: 8.1.12 (10) (World Series odds) Show your work.

Note from Claude: In a 7-game series (such as the real American baseball World Series), the first team to win 4 games wins the series. 7 is the maximum number of games that can be played before one of the teams must win four games. But if one team wins 4 games sooner, the series ends immediately.

10. ▷ *World Series odds* Consider two teams, A and B, playing a series of games until one of the teams wins n games. Assume that the probability of A winning a game is the same for each game and equal to p and the probability of A losing a game is $q = 1 - p$. (Hence, there are no ties.) Let $P(i, j)$ be the probability of A winning the series if A needs i more games to win the series and B needs j more games to win the series.

    a. Set up a recurrence relation for $P(i, j)$ that can be used by a dynamic programming algorithm.

    b. Find the probability of team A winning a seven-game series if the probability of it winning a game is 0.4.

    c. Write a pseudocode of the dynamic programming algorithm for solving this problem and determine its time and space efficiencies.

Problem 2. 8.4.3 [8.2.3] (5) (Warshall with no extra memory use)

Problem 3. 8.4.4[8.2.4] (10) (More efficient Warshall inner loop)

3. Explain how to implement Warshall's algorithm without using extra memory for storing elements of the algorithm's intermediate matrices.

4. Explain how to restructure the innermost loop of the algorithm *Warshall* to make it run faster at least on some inputs.

Problem 4. OptimalBST problem (25 points plus extra credit)

Not from the textbook. Description is in the assignment document

Problem 5. 8.3.3 (10) (Optimal BST from root table)

3. Write a pseudocode for a linear-time algorithm that generates the optimal binary search tree from the root table.

You may do this for the "successful searches only" approach form the Levitin textbook if you prefer.

Problem 6. 8.3.4 (5) (Sum for optimalBST in constant time)

4. Devise a way to compute the sums $\sum_{s=i}^{j} p_s$, which are used in the dynamic programming algorithm for constructing an optimal binary search tree, in constant time (per sum).

Problem 7. 8.3.6 (10) (optimalBST--successful search only--if all probabilities equal)

6. How would you construct an optimal binary search tree for a set of n keys if all the keys are equally likely to be searched for? What will be the average number of comparisons in a successful search in such a tree if $n = 2^k$?

This problem is postponed to a later assignment.

Problem 8. 8.3.10a (5) (Matrix chain multiplication)

10. ***Matrix chain multiplication*** Consider the problem of minimizing the total number of multiplications made in computing the product of n matrices

$$A_1 \cdot A_2 \cdot \ldots \cdot A_n$$

whose dimensions are d_0 by d_1, d_1 by $d_2, \ldots, d_{n-1}$ by d_n, respectively. (Assume that all intermediate products of two matrices are computed by the brute-force (definition-based) algorithm.

a. Give an example of three matrices for which the number of multiplications in $(A_1 \cdot A_2) \cdot A_3$ and $A_1 \cdot (A_2 \cdot A_3)$ differ at least by a factor 1000.

Author's hint, problem 1:

10. a. In the situation where teams A and B need i and j games, respectively, to win the series, consider the result of team A winning the game and the result of team A losing the game.

    b. Set up a table with five rows $(0 \le i \le 4)$ and five columns $(0 \le j \le 4)$ and fill it by using the recurrence derived in part (a).

    c. A pseudocode should be guided by the recurrence set up in part (a). The efficiency answers follow immediately from the table's size and the time spent on computing each of its entries.

Author's hints, problems 2 and 3:

3. Show that we can simply overwrite elements of $R^{(k-1)}$ with elements of $R^{(k)}$ without any other changes in the algorithm.

4. What happens if $R^{(k-1)}[i, k] = 0$?

Author's hint, problem 4:

This problem is not from the Leviton textbook; no author's hints. The details are on the assignment document.

Author's hint, problem 5:

3. $k = R[1, n]$ indicates that the root of an optimal tree is the kth key in the list of ordered keys $a_1, \ldots, a_n$. The roots of its left and right subtrees are specified by $R[1, k-1]$ and $R[k+1, n]$, respectively.

Author's hint, problem 6:

:

4. Use a space-for-time tradeoff.

Author's hint, problem 7:

6. The structure of the tree should simply minimize the average depth of its nodes. Do not forget to indicate a way to distribute the keys among the nodes of the tree.

Author's hint, problem 8:

10. a. It is easier to find a general formula for the number of multiplications needed for computing $(A_1 \cdot A_2) \cdot A_3$ and $A_1 \cdot (A_2 \cdot A_3)$ for matrices A_1 with dimensions d_0-by-d_1, A_2 with dimensions d_1-by-d_2, and A_3 with dimensions d_2-by-d_3 and then choose some specific values for the dimensions to get a required example.