HW 12 textbook problems and hints

Problem 1. 8.4.3 [8.2.3] (5) (Warshall with no extra memory use)

Problem 2. 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.

Author's hints:

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 $R^{(k-1)}[i, k] = 0$?

Problem 3. OptimalBST problem (25 points plus extra credit)

Not from the textbook. Description is in the assignment document

Problem 4. 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.

Author's hint:

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

Problem 5. 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).

Author's hint

4. Use a space-for-time tradeoff.

:

Problem 6. 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 $n = 2^k$?

Author's hint:

> 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.

Problem 7. 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, ..., 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:

> 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.