CSE 6331 – Algorithms – Spring, 2015 – Prof. Supowit

Homework 5 – Due: Monday, February 23

1. Design and analyze the running time of a dynamic programming algorithm for the following problem: given a sequence $(x_1, x_2, \ldots, x_n)$ of integers, find a longest subsequence $(x_{i_1}, x_{i_2}, \ldots, x_{i_k})$ such that

$$\left| x_{i_j} - x_{i_{j+1}} \right| \leq 5 \quad \forall j, 1 \leq j \leq k-1 .$$

2. Consider the recursive (i.e., divide-and-conquer) algorithm that solves the matrix chain multiplication problem by directly using the recurrence relation that we saw in class:

$$m_{i,j} = \begin{cases} 0, & \text{if } i = j \\ \min\limits_{i \leq k < j} \left\{ m_{i,k} + m_{k+1,j} + p_{i-1} p_k p_j \right\}, & \text{if } i < j \end{cases}$$

for each i and j such that $1 \leq i \leq j \leq n$. We called this Alg_1 in class.

Now consider another approach, which we'll call Algorithm Alg_0, that enumerates each possible parenthesization of the given matrix chain and picks the optimal one.

Which is asymptotically faster, Algorithm Alg_1 or Alg_0? Why?

Of course, the dynamic programming algorithm is much, much faster than either *A* or *B*.

3. A greedy approach to the matrix chain problem is as follow:

Find the value of k that minimizes $p_0 p_k p_n$. This gives us the top level of the parenthesization: $(A_1 \times A_2 \times \cdots \times A_k) \times (A_{k+1} \times A_{k+2} \times \cdots \times A_n)$. Then recursively do likewise on the left chain $A_1 \times A_2 \times \cdots \times A_k$ and on the right chain $A_{k+1} \times A_{k+2} \times \cdots \times A_n$.

Does this algorithm guarantee an optimal solution? Prove your answer.

4. For the longest common subsequence problem, write pseudo-code for a top-down, recursive algorithm using memoization. Your code needs to output just the length of a longest common subsequence, rather than an actual subsequence. Analyze the running time of your algorithm, given sequences $X = x_1 x_2 \cdots x_m$ and $Y = y_1 y_2 \cdots y_n$.