CMPS 2200 Introduction to Algorithms – Fall 12

11/19/12

9. Homework

Due **11/28/12** in class

1. **LCS traceback (6 points)**
   (a) Give pseudocode that performs the traceback to construct an LCS from a filled dynamic programming table *with* using the "arrows", in $O(n+m)$ time. *(For an elegant solution you could use recursion to use the recursion stack to reverse the output sequence on the fly.)*
   (b) Give pseudocode that performs the traceback to construct an LCS from a filled dynamic programming table *without* using the "arrows", in $O(n+m)$ time. Justify shortly why your algorithm is correct. *(Hint: You need to essentially "recompute" the information.)*

2. **Binomial coefficient (10 points)**
   Given n and k with $n \geq k \geq 0$, we want to compute the binomial coefficient $\binom{n}{k}$.
   (a) (5 points) Give pseudo-code for the bottom-up dynamic programming algorithm to compute $\binom{n}{k}$ using the recurrence

$$\begin{aligned}\binom{n}{k} &= \binom{n-1}{k-1} + \binom{n-1}{k}, \text{ for } n > k > 0 \\ \binom{n}{0} &= \binom{n}{n} = 1, \text{ for } n \geq 0\end{aligned}$$

   (b) (1 point) What are the runtime and the space complexity of your algorithm, in terms of n and k?
   (c) (4 points) Now assume you use memoization to compute $\binom{4}{2}$ using the above recurrence. In which order do you fill the entries in the DP-table? Give the DP-table for this case and annotate each cell with a "time stamp" (i.e., with a number $1, 2, 3, \ldots$) when it was filled. *(Hint: Draw the recursion tree and fill the table in the order determined by the recursive evaluation.)*

Flip over to back page $\Longrightarrow$

3. **Checkerboard (8 points)**

Suppose that you are given an $n \times n$ checkerboard. A checker is allowed to move from its current square to (1) the square immediately above, (2) the square that is one up and one to the left, and (3) the square that is one up and one to the right, as long as it does not leave the checkerboard. You are also given a cost function c which specifies for every valid move from square (i_1, j_1) to square (i_2, j_2) a possibly negative amount of $c((i_1, j_1), (i_2, j_2))$ dollars.

Your task is to design a dynamic programming algorithm that finds a path from some position on the bottom edge to some position on the top edge of the checkerboard, such that the total dollar amount that is generated by the moves on the path is maximized. *(Hint: The first part of the sentence specifies base cases, the second the recursive case.)*

Define a recurrence relation for the total cost of a path to a given square:

$$\mathit{total}(i, j) = \text{ total cost of a path to square } (i, j),$$

for all i, j. Briefly justify the correctness. Then argue in words (no pseudo-code necessary) how the dynamic programming matrix would be filled and finally traced back to find an optimal path, and give the runtime of the resulting algorithm.