CS 5633 Analysis of Algorithms – Spring 06

3/23/06

7. Homework

Due **Thursday 3/30/06** before class

1. **Binomial coefficient (5 points)**
   Given n and k with $n \geq k \geq 0$, we want to compute the binomial coefficient $\binom{n}{k}$. However, we are only allowed to use additions, and no multiplications.

   a) (2 points) Give a 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, expressed in n *and* k?

   e) (2 point) Now assume you use memoization to compute $\binom{4}{3}$ 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" when it was filled.

2. **LCS traceback (3 points)**
   Show how to perform the traceback in order to construct an LCS from the filled dynamic programming table *without* using the "arrows", in $O(n+m)$ time. Justify your answer.

3. **Matrix chain multiplication traceback (3 points)**
   Show how to perform the traceback in order to construct an optimal parenthesization for the matrix chain multiplication problem *without* using the auxiliary s-table. How much time does the traceback algorithm need? Justify your answer.

4. **Saving space (5 points)**
   Suppose we only want to compute the *length* of an LCS of two strings of length m and n. This means we do not need to store the whole dynamic programming table for a later traceback.

   Show how to alter the dynamic programming algorithm such that it only needs $\min(m,n)+O(1)$ space. (Notice that it is *not* $O(\min(m,n))$, but plain $\min(m,n)$.)

FLIP OVER TO BACK PAGE ⟹

5. **Intervals (8 points)**

1. Let $A[1..n]$ be an array of n integers (which can be positive, negative, or zero). An *interval* with start-point i and end-point j, $i \leq j$, consists of the numbers $A[i], \ldots, A[j]$ and the *weight* of this interval is the sum of all elements $A[i] + \ldots + A[j]$.

   The problem is: Find the interval in A with maximum weight.

   (a) **(2 points)** Describe an algorithm for this problem that is based on the following idea: Try out all combinations of i, j with $1 \leq i < j \leq n$. What is the runtime of this algorithm?

   (b) Describe a dynamic programming algorithm for this problem. Proceed in the following steps:

      i. **(2 points)** Develop a recurrence for the following entity: $S(j) =$ maximum of the weights of all intervals with end-point j.

      ii. **(1 point)** Based on this recurrence describe an algorithm that computes all $S(j)$ in a dynamic programming fashion, and afterwards determines the end-point j^* of an optimal interval.

      iii. **(2 points)** Given the end-point j^* find the start-point i^* of an optimal interval by backtracking.

      iv. **(1 point)** What are the runtime and the space complexity of this algorithm?