CS 5633 Analysis of Algorithms – Spring 08

3/6/08

7. Homework

Due **Thursday 3/13/08** 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 $\binom{4}{2}$ 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 $1, 2, 3, ...$) when it was filled.

2. **Saving space (5 points)**

   a) (2 points) The bottom-up dynamic programming algorithm computing the n-th Fibonacci number $F(n)$ takes $O(n)$ time and uses $O(n)$ space. Show how to modify the algorithm to use only constant space.

   a) (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 this traceback algorithm need? Justify your answer.

3. **LCS for three strings (5 points)**
   Give an algorithm that can find a longest common subsequence of **three** input strings. Your algorithm should be a dynamic programming algorithm that runs in $O(n_1 n_2 n_3)$ time, where n_1, n_2, n_3 are the lengths of the tree strings, respectively. You do not need to formally prove the correctness of the recurrence that you will develop, but please give a convincing argument.

FLIP OVER TO BACK PAGE $\Longrightarrow$

4. **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?

Related questions from previous PhD Exams

Just for your information. You **do not** need to solve them for homework credit.

1. This problem is concerned with recurrences, recursion and dynamic programming.

   Consider the following recurrence:

$$T(n) = \sum_{i=1}^{n} T(i-1)T(n-i)$$

   for any $n > 0$, and $T(0) = 1$. We are concerned with computing $T(n)$ for a given value of $n \geq 1$.

   (a) Calculate the values of $T(1)$, $T(2)$, and $T(3)$.

   (b) Use mathematical induction to prove that $2^n \leq T(n) \leq n!$ for all $n \geq 0$.

   (c) Provide an algorithm that implements the recursion directly and show that it requires an exponential number, in n, of arithmetic operations.

   (d) Give a dynamic programming algorithm that only uses a polynomial number of arithmetic operations, and analyze its runtime.