CSCI-GA.1170-001/002 Fundamental Algorithms	October 29, 2014

Problem Set 7

Lecturer: Yevgeniy Dodis — *Due: Wednesday, November 5*

Problem 7-1 (Text Alignment) — 6 points

Using dynamic programming, find the optimum printing of the text *"Not all those who wander are lost"*, i.e. $\ell_1 = 3, \ell_2 = 3, \ell_3 = 5, \ell_4 = 3, \ell_5 = 6, \ell_6 = 3, \ell_7 = 4$, with line length $L = 14$ and penalty function $P(x) = x^3$. Will the optimal printing you get be consistent with the strategy "print the word on as long as it fits, and otherwise start a new line"? Once again, you have to actually find the alignment, as opposed to only finding its penalty.

Problem 7-2 (Shortest Common Super-sequence) — 20 points

Let $X[1 \dots m]$ and $Y[1 \dots n]$ be two given arrays. A common supersequence of X and Y is an array $Z[1 \dots k]$ such that X and Y are both subsequences of $Z[1 \dots k]$. Your goal is to find the *shortest* common super-sequence (SCS) Z of X and Y, solving the following sub-problems.

(a) (4 points) First, concentrate on finding only the length k of Z. Proceeding similarly to the longest common subsequence problem, define the appropriate array $M[0 \dots m, 0 \dots n]$ (in English), and the write the key recurrence equation to recursively compute the values $M[i,j]$ depending on some relation between $X[i]$ and $Y[j]$. Do not forget to explicitly write the base cases $M[0,j]$ and $M[i,0]$, where $1 \le i \le m, 1 \le j \le n$.

(b) (5 points) Translate this recurrence equation into an explicit bottom-up $O(mn)$ time algorithm that computes the length of the shortest common supersequence of X and Y.

(c) (5 points) Find the SCS of $X = BARRACUDA$ and $Y = ABRACADABRA$. (Notice, you need to find the actual SCS, not only its length.)

(d) (6 points) Show that the length k of the array Z computed in part (a) satisfies the equation $k = m + n - \ell$, where ℓ is the length of the longest common *subsequence* of X and Y.
(**Hint**: Use the recurrence equation in part (a), then combine it with a similar recurrence equation for the LCS, and then use induction. There the following identity is very handy: $\min(a,b) + \max(a,b) = a + b$.)

Problem 7-3 (Babysitting Dillemma) — 18 points

You are a CFO of a baby sitting company, and got a request to baby sit n children one day. You can hire several babysitters for a day for a fixed cost B per babysitter. Also, you can assign an arbitrary number of children $i \ge 1$ to a babysitter. However, each parent will only pay some amount $p[i]$ if his child is taken care of by a babysitter who looks after i children. For example, if $n = 7$ and you hire 2 babysitters who looks after 3 and 4 children, respectively, you revenue is $3p[3] + 4p[4] - 2B$.

Given $B, n, p[1], \ldots, p[n]$, your job is to assign children to babysitters as to maximize your total profit. Namely, you want to find an optimal number k and an optimal partition $n = n_1 + \ldots + n_k$ so as to maximize revenue $R = n_1 \cdot p[n_1] + \ldots + n_k \cdot p[n_k] - k \cdot B$.

(a) (5 points) Let $R[i]$ denote the optimum revenue you can get by looking after i children. E.g., $R[0] = 0$, $R[1] = p[1] - B$, $R[2] = \max(2p[1] - 2B, 2p[2] - B)$, etc. Write a top-down recursive formula for $R[n]$ in terms of values $R[j]$ for $j < n$.

(b) (5 points) Write a top-down recursive procedure with memorization which will compute $R[n]$. Analyze the running time of your procedure in the $\Theta(\cdot)$ notation.

(c) (5 points) Write an iterative bottom-up variant of the same procedure.

(d) (3 points) Explain how to augment your procedure (either in part (b) or (c)) to also compute the optimal number of babysitters k and the actual partition of children. Either English or pseudocode will work.

Problem 7-4 (Dividing Chocolate) 10 points

You have $m \times n$ chocolate bar. You are also given a matrix $\{p[i,j] \mid 1 \le i \le m, 1 \le j \le n\}$ telling you the price of the $i \times j$ chocolate bar. You are allowed to repeat the following procedure any number of times, starting initially with the single big $m \times n$ piece you have. Take one of the pieces you have and split it into two pieces by cutting it either vertically or horizontally. Say, $m = 5$, $n = 4$. You may first choose to split it into two pieces of size 3×4 and 2×4. Then you may take the 3×4 piece and split it into two pieces 3×2 and 1×2. Finally, you may take the previous 2×4 piece and split it into two 1×4 pieces. If you stop, you have four pieces of sizes 3×2, 1×2, 1×4 and 1×4, which you can sell for $p[3,2] + p[1,2] + 2p[1,4]$. You goal is to find a partition maximizing your total profit.

(a) (5 points) Let $C[i,j]$ be the largest profit you can get by splitting an $i \times j$ piece, where $0 \le i \le m$, $0 \le j \le n$ and we set $C[i,0] = C[0,j] = 0$. Write a recursive formula for $C[m,n]$ in terms of values $C[i,j]$, where either $i < m$ or $j < n$.

(b) (5 points) Write a bottom-up procedure to compute $C[m,n]$ and analyze its running time as a function of m and n.