CSCI-GA.1170-003/004 Fundamental Algorithms March 27, 2019

Problem Set 7

Lecturer: Yevgeniy Dodis *Due: Thursday, April 4*

Problem 7-1 (Studying for Finals) 14 Points

Priya has already started to plan a schedule for studying for finals. There are n hours left before Priya's first final, and Priya can study during any of those hours. However, before each hour of studying, Priya needs to first spend b hours procrastinating.

For example, if $n = 5$ and $b = 1$, then Priya has 5 hours to study, and needs to procrastinate for 1 hour before studying for 1 hour. Possible options include, studying for the third and fifth hour, studying for the second hour only, or not studying at all. In this problem, you will help Priya plan a study schedule.

(a) (4 points) Let $B[n]$ be the number of possible schedules for the first n hours. Give with explanation a recurrence relation for $B[n]$ and write explicitly the base case.

(**Hint**: At each hour n Priya can either study or not study. For each option, consider how she can spend her time before hour n.)

(b) (2 points) Give an algorithm that runs in time $O(n)$ to compute the total number of study schedules for Priya.

(c) (4 points) Depending on the hour, Priya might gain more utility from studying. In particular, assume that by studying in hour i, Priya will get $p[i]$ more points on her final.

Let $C[n]$ be the maximum number of points Priya will get on her final if she only studies in the first n hours. Give with explanation a recurrence relation for $C[n]$ and write explicitly the base case.

(d) (4 points) Give an algorithm that prints the hours that Priya should study so that she can get the maximum number of points on her final.

Problem 7-2 (Free-Pizza Collector) 10 Points

Assume that you attend a boring science convention that takes place in a conference center that is arranged as a n by m grid with a mini-exhibition at each node. To attract customers, each exhibition offers a certain number of free pizza slices to passing customers. You want to move from north-west corner to the south-east corner of the conference center. Since you do not want to spend too much time there, in each step you may either move south or east. Your goal is to maximize the total number of free pizza slices you can get.

More formally, if you pass by the exhibition located at node (i, j) where $0 \leq i \leq n$ and $0 \leq j \leq m$, you get $p[i, j]$ slices of pizza. You are told all of the $p[x, y]$ a priori. Your goal is to design a path from $(0, 0)$ to (n, m) allowing you to consume as much of pizza as you can!

Give an efficient (i.e., polynomial in n and m) dynamic programming algorithm for this problem, and analyze its running time.

Problem 7-3 (Shortest Common Super-sequence) 18 points

Let $X[1 \ldots m]$ and $Y[1 \ldots n]$ be two given arrays. A common supersequence of X and Y is an array $Z[1 \ldots k]$ such that X and Y are both subsequences of $Z[1 \ldots k]$. 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 $M[0 \ldots m, 0 \ldots n]$ (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 $1 \leq i \leq m, 1 \leq j \leq n$.

(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) (3 points) Find the SCS of $X = NONE$ and $Y = NOON$ (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: $\min(a, b) + \max(a, b) = a + b$.)