Lahore University of Management Sciences

CMPE 510: Design & Analysis of Algorithms (Spring 2013-2014)

Homework 5

Due: Wednesday, March 19, 2014 (in class)

Note: Collaboration on homeworks is encouraged. However, you should think about the problems yourself before discussing them with others. Furthermore, you must write up your solutions by yourself and understand anything that you hand in. If you do collaborate, you must acknowledge your collaborators in your write-up.

1. You are given n events to be scheduled in one available room. Each event takes one unit of time and only one event can be scheduled in the room at one time. Event i will give a profit of g_i dollars ($g_i > 0$) if started at or before time t_i where t_i is an arbitrary real number. If event i is not started by t_i, there is no benefit in scheduling it at all. All events can start as early as time 0.

   Give a greedy algorithm to find a schedule that maximizes the profit and prove that your algorithm is correct. (*Hint:* Consider which event to schedule in the last slot and then second last slot and so on.)

2. Suppose we want to make change for n cents, using the least number of coins of denominations 1, 10, and 25 cents. Describe an $O(n)$ dynamic programming algorithm to find an optimal solution. (There is also an easy $O(1)$ algorithm but the idea here is to illustrate dynamic programming).

3. You are travelling by canoe down a river and there are n trading posts along the way. Before starting your journey, you are given for each $1 \leq i < j \leq n$, the fee f_{ij} for renting a canoe from post i to post j. These fees are arbitrary. For example it is possible that $f_{1,3} = 10$ and $f_{1,4} = 5$. You begin at trading post 1 and must end at trading post n (using rented canoes). Your goal is to minimize the rental cost. Design an efficient algorithm for the problem analyze its running time.

4. **Longest Increasing Subsequence:** Given an array of numbers $A[1\,..\,n]$ a *subsequence* is any subset of these numbers taken in order, of the form $A[i_1], A[i_2], \ldots, A[i_k]$ where $1 \leq i_1 < i_2 < \cdots < i_k \leq n$, and an *increasing subsequence* is one in which the numbers are getting strictly larger. The *longest increasing subsequence* is the increasing subsequence of greatest length. E.g. the longest increasing subsequence of [5, 2, 8, 6, 3, 6, 9, 7] is 2, 3, 6, 9 of length 4.

   Design a dynamic programming formulation to solve the longest increasing subsequence problem. (*Hint:* Define OPT(i, j) as the length of the longest increasing subsequence of $A[j\,..\,n]$ with all elements greater than $A[i]$.)

5. A certain string-processing language allows the programmer to break a string into two pieces. Since this involves copying the old string, it costs n units of time to break a string of n characters into two pieces. Suppose a programmer wants to break a string into many pieces. The order in which the breaks are made can effect the total amount of time used. For example, suppose we wish to break a 20 character string after characters 3, 8, and 10 (numbering the characters in ascending order from the left-hand side, starting from 1). If the breaks are made in the left-to-right order, then the first break costs 20 units of time, the second break

costs 17 units of time, and the third break costs 12 units of time, a total of 49 units of time. If the breaks are made in the right-to-left order, then the first break costs 20 units of time, the second break costs 10 units of time, and the third break costs 8 units of time, a total of 38 units of time.

(a) Devise a dynamic programming algorithm that, when given the indices of the charcters after which to break, determines the cheapest cost of those breaks in time $O(n^3)$.

(b) Find a counterexample to the following algorithm for the problem above.
Start by cutting the string as close to the middle as possible, and then repeat the same thing recursively in each half.

6. [*]In the 5 by 5 matrix below, the minimal path sum from the top left to the bottom right, by *only moving to the right and down*, is indicated in bold and is equal to 2427.

$$\begin{bmatrix} \mathbf{131} & 673 & 234 & 103 & 18 \\ \mathbf{201} & \mathbf{96} & \mathbf{342} & 965 & 150 \\ 630 & 803 & \mathbf{746} & \mathbf{422} & 111 \\ 537 & 699 & 497 & \mathbf{121} & 956 \\ 805 & 732 & 524 & \mathbf{37} & \mathbf{331} \end{bmatrix}$$

Use your favorite programming language to find the minimal path sum, in matrix.txt[†], a text file containing a 80 by 80 matrix, from the top left to the bottom right by only moving right and down. (*Note:* Code submission guidelines will be posted later.)

7. [‡]The minimal path sum in the 5 by 5 matrix below, by starting in any cell in the left column and finishing in any cell in the right column, and only moving up, down, and right, is indicated in bold; the sum is equal to 994.

$$\begin{bmatrix} 131 & 673 & \mathbf{234} & \mathbf{103} & \mathbf{18} \\ \mathbf{201} & \mathbf{96} & \mathbf{342} & 965 & 150 \\ 630 & 803 & 746 & 422 & 111 \\ 537 & 699 & 497 & 121 & 956 \\ 805 & 732 & 524 & 37 & 331 \end{bmatrix}$$

Use your favorite programming language to find the minimal path sum, in matrix.txt[§], a text file containing a 80 by 80 matrix, from the left column to the right column and only moving up, down, and right. (*Note:* Code submission guidelines will be posted later.)

[*]Problem 81 on Project Euler.
[†]See attachments with Homework 5 on LMS
[‡]Problem 82 on Project Euler.
[§]See attachments with Homework 5 on LMS