CSE5311 Design and Analysis of Algorithms
Fall 2009
September 28, 2009
Exercise Set 5

1. Suppose you are managing the construction of billboards on the Stephen Daedalus Memorial Highway, a heavily traveled stretch of road that runs west-east for M miles. The possible sites for billboards are given by numbers $x_1, x_2, \ldots, x_n$, each in the interval $[0,M]$ (specifying their position along the highway measured in miles from its western end). If you place a billboard at location x_i, you receive a revenue of $r_i > 0$. Regulations imposed by the county's Highway Department require that no two billboards be within less than or equal to 5 miles of each other. You'd like to place billboards at a subset of sites so as to maximize your total revenue, subject to this restriction.

2. Design a dynamic programming algorithm for the ***change-making problem***: given an amount n and unlimited quantities of coins of each of the denominations d_1, d_2, ... d_m, find the smallest number of coins that add up to n or indicate that the problem does not have a solution.

3. Suppose we want to replicate a file over a collection of n servers, labeled $S_1, S_2, \ldots, S_n$. To place a copy of the file at server S_i results in a placement cost of c_i, where c_i is an integer greater than 0. Now if a user requests the file from server S_i, and no copy of the file is present at S_i, then servers $S_{i+1}, S_{i+2}, \ldots, S_{i+3} \ldots$ are searched in order until a copy of the file is finally found, say at server S_j, where $j > i$. This results in an access cost of $j-i$. (Note that the lower-indexed servers S_{i-1}, S_{i-2} … are not consulted in this search.) The access cost is 0 if Si holds a copy of file. We will require that a copy of the file be placed at server S_n, so that all such searches will terminate, at the latest at S_n.

   We'd like to place copies of the files at servers so as to minimize the sum of placement and access costs. Formally, we say that a configuration is a choice, for each server S_i with i = 1, 2, …n-1, of whether to place a copy of the file at S_i or not. (Recall that a copy is always placed at S_n.) The total cost of a configuration is the sum of all placement costs for servers with a copy of the file, plus the sum of all access costs associated with all n servers. Give a dynamic programming solution to find a configuration of minimum total cost.

4. Show how to compute the length of an LCS using only 2 min(m,n) entries in the c table plus O(1) additional space. Then, show how to do this using min(m,n) entries plus O(1) additional space.

5. A child wants to construct the tallest tower possible out of building blocks. She has n types of blocks, and an unlimited supply of blocks of each type. Each type-i block is a rectangular solid with linear dimensions $<x_i, y_i, z_i>$. A Block can be oriented so that any two of its three dimensions determine the dimensions of base and the other dimension is the height. In building a tower one block may be placed on top of another block as long as the two dimensions of the base of the upper block are each strictly smaller than the corresponding base dimensions of the lower block. (Blocks oriented to have equal-sized bases cannot be stacked). Design a dynamic programming algorithm to determine the tallest tower the child can build.

6. Let $R(i,j)$ be the number of times that table entry m[i,j] is referenced by Matrix-Chain order in computing other table entries. Show that the total number of references for the entire table is given by,

$$\sum_{i=1}^{n} \sum_{j=1}^{n} R(i,j) = \frac{n^3 - n}{3}$$

7. Suppose you have n video streams that need to be sent, one after another, over a communication link. Stream i consists of a total of b_i bits that need to be sent, at a constant rate, over a period of t_i seconds. You cannot send two steams at the same time, so you need to determine a schedule for the streams: an order in which to send them. Whichever order you choose, there cannot be any delays between the end of one stream and the start of the next. Suppose your schedule starts at time 0 (and therefore ends at $\sum_{i=1}^{n} t_i$, whichever order you choose.) We assume that all the values b_i and t_i are positive integers. A schedule is valid if it satisfies the constraint imposed by the link: for each natural number $t > 0$, the total number of bits you send over the time interval from 0 to t cannot exceed $r \times t$, where r is the link parameter in bits/sec.
   a. Claim: There exists a valid schedule if and only if each stream i satisfies $b_i \leq r \times t_i$. Decide whether the claim is true or false, and give a proof of either the claim or its negation.
   b. Give an algorithm that takes a set of n streams, each specified by its number and bits b_i and its duration t_i as well as the link parameter r_i and determines whether there exists a valid schedule. The running time of your algorithm should be polynomial in n.

8. A ski rental agency has m pairs of skis, where the height of the ith pair of skis is s_i. There are n skiers who wish to rent skis, where the height of the ith skier is h_i. Ideally, each skier should obtain a pair of skis whose height matches with his own height as closely as possible. Design and efficient algorithm to assign skis so that the sum of the absolute differences of the heights of each skier and his/her skis is minimized.

9. Consider the problem of neatly printing a paragraph on a printer. The input text is a sequence of n words of length $l_1, l_2, \ldots, l_n$, measured in input characters. We want to print this paragraph neatly on a number of lines that hold a maximum of M characters each. Our criterion of "neatness" is as follows. If a given line contains words i through j and leave exactly one space between words, the number of extra space characters at the end of the line is

$$M - j + i - \sum_{k=i}^{j} l_k$$

   We wish to minimize the sum, over all the lines except the last of the extra space characters at the ends of lines. Give a dynamic programming algorithm to print a paragraph of n words neatly on a printer. Analyze the running time and space requirements of your algorithm.