Fall 2001 **CMSC 451: Homework 4** Samir Khuller

Due in class: 10am Oct 23.

This homework is shorter as you have to prepare for the exam as well. No late homeworks will be accepted since solutions will go out on Oct 23 itself. If you forget to bring your HW to class, I will NOT take it after 10am.

(1) Design an algorithm that runs in time $O(nK)$ that takes as input a set $S = \{x_1, x_2, \ldots, x_n\}$ of integers, and a value K and returns a subset S' of S that adds exactly to K. In other words, if such a subset exists, your algorithm should output it.

(2) Design an $O(nK)$ algorithm for the integer Knapsack problem. Given a set of n items, where the weight of the ith item is w_i and its profit is p_i, find a maximum profit subset of items that adds to *at most* K. In other words, each item can is in or out. We have to select a subset of items that yield the maximum profit, and whose weight does not exceed K. Your algorithm should output such a subset as well.

(3) We are going on a trip along the Appalachian trail. We have a list of all possible campsites that we can camp in along the way (say n). We want to do this trip in exactly K days, stopping $K-1$ nights to camp. Our goal is to plan this trip so that we minimize the maximum amount of walking done in a single day. In other words, if our trip involves 3 days of walking, and we walk 11, 14, 12 miles on each day respectively, the cost is 14. Another schedule that involves walking 11, 13, 13 miles on each day has cost 13. Note that the location of the campsites is specified in advance, and we can only campat a campsite. Your algorithm should be based on dynamic programming and run efficiently (polynomial time). For full credit, obtain a time bound of no more than $O(nK)$. The algorithm should output the distance d, such that on each day we walk at most d miles. Also, d should be the smallest possible value with this property. For graduate students: come up with a non-dynamic programming algorithm as well!

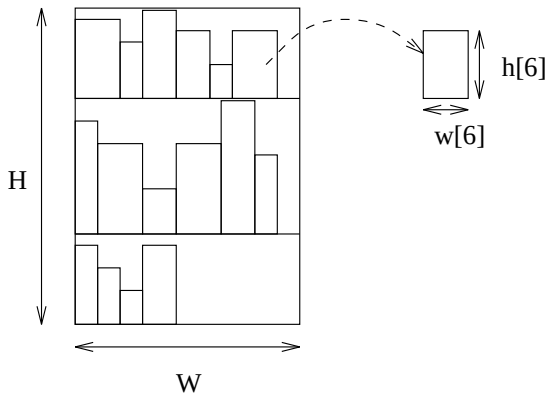
(4) We are given a collection of n books, which must be placed in a library book case. Let $h[1..n]$ and $w[1..n]$ be arrays, where $h[i]$ denotes the height of the i-th book and $w[i]$ denotes the width of the i-th book. Assume that these arrays are given by the books' catalogue numbers (and the books MUST be shelved in this order), and that books are stacked in the usual upright manner (i.e., you CANNOT lay a book on its side). All book cases have width W, but you may have any height you like. The books are placed on shelves in the book case. You may use as many shelves as you like, placed wherever you like (up to the height of the book case), and you may put as many books on each shelf as you like (up to the width of the book case). Assume for simplicity that shelves take up no vertical height.

The *book-shelver's problem* is, given $h[1..n]$, $w[1..n]$, and W, what is the minimum height book case that can shelve all of these books. Below shows an example of a shelving. The height and width of the 6th book are shown to the right. (Notice that there is some unnecessarily wasted height on the third shelf.)

(a) Write down a recurrence for book shelver's problem.

(b) Use your recurrence to develop an efficient dynamic programming for solving the book shelver's problem.

(c) Analyze the running time of your algorithm.


h[6]
w[6]
H
W