CS 332 Homework Assignment 5

Spring, 2005

Problem 1. Suppose that we are given the following instance of the 0-1 integer knapsack problem.

Item	**Weight**	**Value**
1	3	$25
2	2	$25
3	1	$15
4	4	$40
5	5	$50

And the capacity of the knapsack is W = 6.

Part (a): Fill out, by hand, the table defined by the recurrence relation

$$KV[i,j] = \begin{cases} KV[i-1,j] & j < w[i] \\ \max(KV[i-1,j],\ v[i] + KV[i-1,j-w[i]]) & w[i] \le j \end{cases}$$

where $KV[i,j]$ is the optimal "knapsack value" for a knapsack of capacity j using only items from 1 to i.

Part (b): How many optimal solutions does this instance of the problem have, that is, how many different ways can the knapsack be filled to the optimal value? Show how to use the table from Part (a) to find all of the optimal solutions.

Problem 2. A robot can take steps of 1 meter, 2 meters, or 3 meters. We want to find an algorithm that computes the number of ways the robot can walk n meters, with the order of steps taken into account.

Part (a): Write a recurrence relation for a dynamic programming algorithm. Be sure to include suitable boundary conditions.

Part (b): Find the number of ways that the robot can walk 100 meters. (In other words, implement the dynamic programming algorithm.)

Problem 3. The two-dimensional 0-1 integer knapsack problem is defined as follows. Maximize $\sum_{i=1}^{n} x_i p_i$ subject to the constraints $\sum_{i=1}^{n} x_i w_i \leq c$ and $\sum_{i=1}^{n} x_i v_i \leq d$ where each x_i is from { 0, 1 }, 1<= *i* <= *n*. Write down a recurrence relation for a dynamic programming algorithm for this problem. Be sure to include proper boundary conditions. (Think of the w_i as weights and the v_i as volumes. So the knapsack constrains both the weight and volume of what is put in it.)

Problem 4: A machine has *n* components. For each component there are three suppliers. The weight of component *i* from supplier *j* is $W_{i,j}$ and its cost is $C_{i,j}$ with $j = 1, 2, 3$. The cost of the machine is the sum of the component costs, and its weight is the sum of the component weights. The problem is to determine from which supplier to buy each component from so as to have the lightest machine with cost no more than *c*.

Part (a): Write a dynamic programming recurrence relation for a table *w[i,j]* where *w[i,j]* is the least-weight machine, composed of components 1 through *i*, that costs no more than *j*.

Part (b): Write a dynamic programming recurrence relation for a table *w[i,j]* where *w[i,j]* is the least-weight machine, composed of components *i* through *n*, that costs no more than *j*.

Problem 5: Give an $O(n^2)$ algorithm to find the maximum contiguous sum in a sequence of *n* real numbers. (Note: The dynamic programming algorithm that we gave in class is $O(n)$.)

Problem 6: A frequent error in written text is the transposition of adjacent letters, as in "witner" instead of "winter". Using two replaces, or one delete and an insert, assigns a cost of 2 to this error. To account for its frequency, we want to assign a cost of 1 to this error. We do this by allowing an additional edit operation:

Transpose two adjacent letters.

Rewrite the recurrence relation for computing edit distance to account for this new operation.
Hint: Look two cells back.