

## Homework 6

Due Wednesday, 2/29/12

Remember: submit each problem, including extra credit, on *its own page*.

For any problem in which you give a dynamic programming algorithm as a solution, provide the following:

- A description of the subproblems,
- A recurrence that relates the subproblems,
- A short justification that the recurrence is correct,
- Pseudocode showing how to use the recurrence to compute the values of the subproblems efficiently,
- Pseudocode showing how to perform a traceback to find the optimal solution, and
- A short runtime analysis of the algorithm (both the computation of the values and the traceback).

## Problems

1. (15 pts) Chapter 6, Exercise 2(b) (page 314). For full credit, your algorithm should have a time complexity of  $O(n)$ . Note: the book asks for an algorithm returning just the value of the optimal plan. **For the homework, we are asking for something different: the actual plan itself.**
2. (15 pts) Consider the following problem. The input is a Directed Acyclic Graph (DAG)  $G = (V, E)$ . Let the vertices  $V$  be labeled  $v_1, v_2, \dots, v_n$  such that if the directed edge  $(v_i, v_j)$  is in  $E$  then  $i < j$ . (Recall from Chapter 3 that  $v_1, v_2, \dots, v_n$  is called a *topological ordering*, which exists if and only if  $G$  is a DAG.) Design a dynamic programming algorithm that outputs the longest path ending at node  $v_n$ , the last node in the topological order. For full credit, your algorithm should have a time complexity of  $O(m + n)$ .
3. (10 pts) In the Bin Packing problem, the input is a set of items  $\{1, 2, \dots, n\}$ , a positive integer *weight*  $w_i$  for each item  $i$ , a positive integer *bin capacity*  $B$ , and a positive integer  $k$ . A YES instance is one in which there is a partition of  $\{1, 2, \dots, n\}$  into  $k$  subsets  $S_1, S_2, \dots, S_k$  such that the sum of the weights of the items in each set  $S_j$  is less than or equal to  $B$ . (You can think of each set  $S_j$  as a bin with capacity  $B$ ; the problem is to determine whether it is possible to pack all  $n$  items into  $k$  bins without overflow.)

For example, a YES instance to the problem is given by  $n = 5$ ,  $B = 4$ ,  $k = 3$ , and

$$(w_1, w_2, w_3, w_4, w_5) = (2, 2, 1, 3, 3) \text{ ,}$$

since in the partition  $S_1 = \{1, 2\}$ ,  $S_2 = \{3, 4\}$ ,  $S_3 = \{5\}$ , the sum of the weights in each set  $S_j$  is less than or equal to 4. However, if we change the weight  $w_3$  from 1 to 2, the instance becomes a NO instance, since there is no such partition.

Prove that Bin Packing is in NP by describing a verifier for the problem. Make sure you describe what the certificates represent and argue that the verifier runs in polynomial time.

## Extra Credit

1. (10 pts) Consider the following problem. The input is a binary tree  $T = (V, E)$  with each leaf  $\ell$  labeled by a symbol  $s_\ell$  from some fixed set  $A$ , as well as a two-dimensional non-negative array of costs indexed by pairs of elements of  $A$  such that  $c[s, t]$  is the cost of changing symbol  $s$  to symbol  $t$  and this satisfies  $c[s, s] = 0$  and  $c[s, t] = c[t, s]$  for all  $s, t$  in  $A$ . Design an algorithm to label each internal node  $v$  of the binary tree by a symbol  $s_v$  from  $A$  so as to minimize the sum over all  $(u, v)$  in  $E$  of  $c[s_u, s_v]$ . This kind of problem arises in computational biology when one wishes to assess a potential evolutionary tree and reconstruct properties of potential ancestral species given this tree. In this case each “symbol” might be a very short sequence—or even just a letter—of DNA or protein that might at a given position within a longer sequence, and  $c[s, t]$  is the cost of the evolutionary change in going from  $s$  to  $t$ .