

# CMPS 102 Homework 6, Spring '05

3 problems, 30 points, due Tuesday May 17th

Reading: Chapter 15 (Dynamic programming)

1. (10 pts) Let  $B(n)$  be the number of different binary search trees containing the  $n$  keys  $\{1, 2, \dots, n\}$ . For this problem you will develop a dynamic programming algorithm that, given  $n$ , calculates  $B(n)$ .

Note that  $B(1) = 1$  since there is only one one-node binary tree. For two nodes  $B(2) = 2$  (either node can be root, and there is only one binary search tree consistent with each choice).

- (a) (1 pt) Calculate by hand  $B(3)$ .
- (b) (1 pt) How many  $n = 6$  key binary trees with key 3 at root (so the left subtree has two nodes, and the right one has 3 nodes) are there?
- (c) (3 pts) Construct a recurrence for  $B(n)$ , I expect something like

$$B(n) = \sum_r \text{number of } n\text{-node trees with root } r,$$

where you need to figure out how many trees there are with root  $r$  in terms of  $B(j)$  values with  $j < n$ . Include boundary conditions in your recurrence; what is a convenient boundary value for  $B(0)$ ?

- (d) (4 pts) Give an iterative (bottom up) algorithm based on the recurrence that computes  $B(n)$  by filling in a table.
  - (e) (1 pt) What is the running time of your iterative algorithm as a function of  $n$  (use  $\Theta$  notation)?
2. (10 pts) Assume you are planning a canoe trip down a river. The river has  $n$  trading posts numbered 1 to  $n$  going downstream. You will start your trip at trading post number 1 and end at trading post number  $n$ . Let  $R(i, j)$  be the cost of renting a canoe at trading post  $i$  and returning it at trading post  $j$ , where  $j > i$ . Assume that you always want to go down river, so the costs if  $j \leq i$  are irrelevant. Find the cheapest sequence of rentals that allow you to complete your trip. Aim for an algorithm running in  $O(n^2)$  time.

For example, if  $n = 4$  and the costs are:

R(i,j)	$j$		
$i$	2	3	4
1	15	25	35
2	--	12	16
3	--	--	5

then the cheapest sequence of canoe rentals to travel the river would be to rent from 1 to 3, and then from 3 to 4 for a cost of  $25 + 5 = 30$ .

On the other hand, if the costs were:

R(i,j)	$j$		
$i$	2	3	4
1	20	15	30
2	--	5	10
3	--	--	20

then taking one canoe all the way from 1 to 4 and renting 1 to 2 and then 2 to 4 are the cheapest solutions (both cost 30). (Note that renting from 1 to 3 is cheaper than going 1 to 2, but the 1 to 3 rental is not in any of the cheapest 1 to 4 solutions.)

Let  $C(k)$  be the cost of the cheapest sequence of canoe rentals starting from trading post 1 and returning the last canoe rented at trading post  $k$ .

First, assume an optimal sequence of rentals changes canoes at some trading post  $j$ . What subproblems are also solved optimally by (parts of) this rental sequence? Prove your answer.

Second, derive a recurrence for  $C(k)$  in terms of  $C(j)$  values where  $j < k$ .

Third (turn in only if you do not get the final part), use your recurrence to construct a recursive algorithm for computing the  $C(k)$  values.

Fourth, give a bottom-up iterative algorithm for computing the  $C(j)$  values.

Finally, show how keeping a little additional information allows the a cheapest sequence of canoe rentals to be printed out.

3. (10 pts) Coins and Change. Assume that you are working a cash register in a country that has strange values for its coins (for example: 1, 5, 8, and 10) and you have to give some amount (like 16) in change using the fewest possible coins. For sensible systems, the greedy method works (take as many of the largest coin as possible without exceeding the amount, go the next largest coin, etc.). However, with this example it is better to take two coins of value 8 than a 10, 5, and 1. For this problem you will develop a dynamic programming algorithm for the Coins and Change problem.

Formally, the Coins and Change problem is:

Given  $n$ , a list of  $k$  integer coin values  $v_1 = 1 < v_2 < v_3 < \dots < v_k$  and an amount  $a$ , find a way to give value exactly  $a$  using the fewest possible coins. Assume that you have plenty of coins of each denomination.

Thus a potential solution is vector  $N = (n_1, n_2, \dots, n_k)$  where each  $n_i$  is non-negative and indicates how many value  $v_i$  coins to give. The solution  $N$  is feasible if  $\sum_{i=1}^k n_i v_i = a$ . The solution  $N$  is optimal if  $N$  is feasible and for every feasible solution  $N' = (n'_1, \dots, n'_k)$  the  $\sum_{i=1}^k n_i \leq \sum_{i=1}^k n'_i$ .

Before proceeding, make sure you understand the problem. Note that the requirement that  $v_1 = 1$  ensures that it is possible to give change for any integer amount.

First, assume we have an optimal solution  $N$ . Find a first or last choice made by the solution and use this first or last choice to identify other (smaller) Coins and Change problems that optimally solved by (part of) the solution  $N$ . Prove that these smaller problems are indeed solved optimally by (parts of) the optimal solution.

Second, focus simply on the number of coins in optimal solutions. Use the insight from the first part to derive a recurrence for the number of coins in optimal solutions.

Third, sketch a top-down recursive algorithm to compute the number of coins in optimal solutions.

Fourth, describe what kind of table can be used to memoize your recursion so that repeated subproblems do not need to be recomputed.

Next, give pseudo-code for an iterative bottom-up algorithm that fills in the table.

Finally, indicate how storing some additional information can allow an optimal solution (the vector  $N$ ) to be constructed.