FALL 2008: **COT 5407** Intro. to Algorithms

[Homework 5; Due Nov 18 at start of class]

General submission guidelines and policies: Add the following signed statement. Without this statement, your homework will not be graded.

> I have adhered to the collaboration policy for this class. In other words, everything written down in this submission is my own work. For problems where I received any help, I have cited the source, and/or named the collaborator.

Read the handout on **Homework guidelines and collaboration policy**.

Problems

Note that all solutions to dynamic programming problems must show (a) the hierarchy of subproblems required to solve the problem, (b) the recurrence relation connecting the solutions of these subproblems along with some explanation for it, (c) the description of the data structure that will store solutions to previously solved subproblems, (d) the actual algorithm, and (e) the time complexity analysis.

23. (**Regular**) Solve Exercise 14.2-2, page 310.

24. (**Exercise**) Solve 14.2-3, page 310.

25. (**Exercise**) Solve 16.3-2, page 392.

26. (**Exercise**) Solve 15.4-1, page 355.

27. (**Regular**) Give an explicit example that shows that the greedy algorithm of picking the item with the highest value to weight ratio is not the best strategy to solve the 0-1 knapsack problem discussed in class.

28. (**Exercise**) Read and understand the application described in Sections 15.1, 15.2, and 15.5.

29. (**Regular**) In class, we discussed greedy and dynamic programming algorithms to solve the *activity selection problem*. Consider the following modification to the problem. Assume that you are going to be paid a bonus of v_i dollars, if you scheduled activity a_i. First show that a greedy algorithm will fail to find the optimal solution. Then design a dynamic programming algorithm to find a set of non-overlapping activities that would leave you with the largest possible bonus. Before you solve this problem, read the **Note** I have added above just before Problem 23.

30. (**Regular**) It is the hurricane season. You have just bought a wooden board of length L (and standard width), which needs to be cut into n smaller pieces so that you can put it up as a hurricane shutter. The cuts are required to be at at locations $l_1, l_2, \ldots, l_n$ ft from the left end of the board. However, the store charges money for cutting. Their cutting rates are strange; if you cut a board of length x into smaller pieces (of any lengths), you will be charged $\$x$.

    The cutting order will determine the cost of the cuttings required. For example, assume that your board is of length 10 ft and that you need to cut it at locations 2ft, 4ft and 7ft from the left end. If you cut it in that order, then your cost will be $\$10 + 8 + 6 = \$24$. On the other hand, if you cut it at 4ft first and then at 2ft and 7ft for the two smaller pieces, then the total cost will be only $\$10 + 4 + 6 = \$20$.

    Design an algorithm to determine the optimal order of cuts required to minimize your total cutting costs. Analyze your algorithm.