CS 473: Algorithms, Fall 2009
HW 7 (due Tuesday, October 27th in class)

This homework contains three problems. **Read the instruction for submitting homework on the course web page**. In particular, *make sure* that you write the solutions for the problems on separate sheets of paper. Write your name and netid on each sheet.

Collaboration Policy: For this home work, students can work in groups of up to 3 members each. Each group submits only one written solution (some groups will do an oral presentation. Indicate your group members on the homework (netids are needed)).

1. (30 pts) Problem 6.16 from the text book.
2. (35 pts) Problem 6.21 from the text book. Modify the algorithm that computes the optimum solution *value* to use only $O(n)$ space.
3. (35 pts) Recall the Knapsack problem discussed in class. The input consists of a knapsack weight limit $W > 0$ and n items; item i has a weight $w_i > 0$ and a value $v_i \geq 0$. The goal is to pack a maximum value subset of the items into the knapsack. Assume that all the input numbers are integers. We described an $O(nW)$ dynamic programming based algorithm. Here we explore a few variants.
   - (5 pts) Suppose $v_i = v$ for all i. Describe an $O(n \log n)$ time algorithm to find an optimum solution. No need to give a proof.
   - (10 pts) Suppose $v_i \in \{a_1, a_2, \ldots, a_k\}$ for $1 \leq i \leq n$ where $a_1, a_2, \ldots, a_k$ are distinct integers; that is, there are only k different values in the given item set. Describe an $O(n^{k+1})$ time algorithm to find an optimum solution. *Hint:* "Guess" the number of items (in other words try all possibilities) chosen from each value a_j and use the idea from the previous part.
   - (20 pts) Let $V = \sum_{i=1}^{n} v_i$ be the maximum possible value that can be obtained. Obtain an $O(nV)$ time dynamic programming based algorithm to find the optimum solution value. For this problem you only need to write a recurrence and justify the number of subproblems and the running time. No need to give an algorithm. If all v_i are integers in the range 1 to 100, what is the running time of the algorithm? *Hint:* For the recurrence, instead of optimizing the value, optimize/minimize the capacity needed to obtain a given value v from the first i items.
   - Extra Credit (10 pts). Suppose all v_i are integers between α and $\alpha + k$ where $\alpha \geq 0$. Describe an algorithm for this case that runs in time polynomial in n and k and return the optimum solution. Note that α can be exponential in n and hence you cannot use the previous part directly.