Solving Knapsack using Dynamic Programming

(25) Problem 8. *Problem 16.2-2 in the text (page 384).*

Consider item i, suppose we have the optimal solutions of the first $i-1$ items (using subsets of items 1 to $i-1$). Now we need to decide the optimal solutions with item i. We defined the symbols as follows:

v_i: the value of the ith item;
w_i: the weight of the ith item;
W: the maximun weight the knapsack can hold;
$F(i,w)$: the maximum value you can get with a knapsack of weight exactly w and a subset of items 1 to i ($F(i,w) = 0$ if no subset has value w).

$F(i,w)$ is an n by W array. Note that we assume the weights are integers.

Now we can either take item i or not. We make our decision on the value we can get in each way, and take the way in which we can get more value.

If we take item i,

$$F(i,w) = F(i-1, w-w_i) + v_i$$

If we do not take item i,

$$F(i,w) = F(i-1,w)$$

Then we get :

$$F(i,w) = max\{\, F\ (i-1,w)F(i-1,w-w_i) + v_i \tag{1}$$

We can easily fill in the top row (where $i = 1$) since we get $F(1,w_1) = v_1$ and all other values are 0. It is then easy to fill in each successive row using the the above formula and the values from the prior row. Since the table has nW entries and each can be computed in $O(1)$ time, the total time is $\Theta(nW)$. The best solution is the largest value in the bottom row, and the actual set can be easily extracted in $\Theta(n)$ time from the table using the same technique as was done using subset sum.

The space is nW but can be easily reduced to $\mathrm{O}(W)$ by keeping only one row of the table and the last item added to get the current value in a table cell.