Week 7 Preparation

(Solutions)

Useful advice: The following solutions pertain to the preparation problems. You are strongly advised to attempt the problems thoroughly before looking at these solutions. Simply reading the solutions without thinking about the problems will rob you of the practice required to be able to solve complicated problems on your own. You will perform poorly on the exam if you simply attempt to memorise solutions to these problems. Thinking about a problem, even if you do not solve it will greatly increase your understanding of the underlying concepts.

Problems

Problem 1. Recall the unbounded knapsack problem shown in the Seminar and notes, where we attempt to find the maximum value that we can fit in a knapsack of capacity C.

Consider the following items with weights and values.

Item i	1	2	3	4	5
Weight w_i	2	3	5	7	10
Value v_i	120	200	250	450	750

For a knapsack with capacity $C = 15$, use the following recurrence relation to fill the memoisation table MaxValue$[c]$ for $0 \leq c \leq 15$.

$$\text{MaxValue}[c] = \begin{cases} 0 & \text{if } c < w_i \text{for all } i, \\ \max\limits_{\substack{1 \leq i \leq n \\ w_i \leq c}} (v_i + \text{MaxValue}[c - w_i]) & \text{otherwise.} \end{cases}$$

where

$$\text{MaxValue}[c] = \{\text{The maximum value that we can fit in a capacity of } c\}$$

Solution

As the question is asking for the entire table we will adopt a bottom up approach to filling out the memoisation table, this just means we will derive the values of MaxValue[0] then MaxValue[1] and so on up to MaxValue[15].

According to the recurrence relation, specifically the first case, if no item can fit into a knapsack for a given capacity c then MaxValue$[c] = 0$. Since the lightest item is 2 then MaxValue$[c] = 0$ for $0 \leq c < 2$.

Capacity	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MaxValue	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Let's now walk through solving MaxValue[2]. Breaking down the second case of the recurrence relation it reads "For each item i whose weight is below c, take choice i where Value$_i$ + MaxValue$[c - \text{Weight}_i]$ is maximised". Now considering MaxValue[2], for each valid item i such that Weight$_i \leq 2$.

1. $120 + \text{MaxValue}[2-2] = 120$
2. Cannot be considered as $w_2 > 2$
3. Cannot be considered as $w_3 > 2$
4. Cannot be considered as $w_4 > 2$
5. Cannot be considered as $w_5 > 2$

Therefore, its optimal to take item 1. Therefore, MaxValue$[2] = 120$.

For MaxValue[3] we now have the option to take item 2 since its weight is 3. Following on what we did before we have

1. $120 + \text{MaxValue}[3-2] = 120$
2. $200 + \text{MaxValue}[3-3] = 200$
3. Cannot be considered as $w_3 > 3$
4. Cannot be considered as $w_4 > 3$
5. Cannot be considered as $w_5 > 3$

Therefore, its optimal to take item 2. Therefore, $\text{MaxValue}[3] = 200$.

The memoisation table now looks as such

Capacity	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MaxValue	0	0	120	200	-	-	-	-	-	-	-	-	-	-	-	-

Using the method above we compute up to the following

Capacity	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MaxValue	0	0	120	200	240	320	400	450	520	-	-	-	-	-	-	-

Let's now walk through solving MaxValue[9]. Again for each item we consider whether it can be taken, and if it can, what is the maximum value knapsack we can derive based on previously computed maximal knapsack's (with lower capacities).

1. $120 + \text{MaxValue}[9-2] = 570$
2. $200 + \text{MaxValue}[9-3] = 600$
3. $250 + \text{MaxValue}[9-5] = 490$
4. $450 + \text{MaxValue}[9-7] = 570$
5. Cannot be considered as $w_5 > 9$

In this case its optimal to take item 2. Therefore, $\text{MaxValue}[9] = 600$.

Repeating the above logic we get the following memoisation table

Capacity	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
MaxValue	0	0	120	200	240	320	400	450	520	600	750	750	870	950	990	1070

Problem 2. Recall the coin change problem described in the Seminar.

$$\text{MinCoins}[v] = \begin{cases} 0 & \text{if } v = 0, \\ \infty & \text{if } v > 0 \text{ and } v < c[i] \text{ for all } i, \\ \min\limits_{\substack{1 \le i \le n \\ c[i] \le v}} (1 + \text{MinCoins}[v - c[i]]) & \text{otherwise.} \end{cases}$$

where

$$\text{MinCoins}[v] = \{\text{The minimum number of coins that we can use to add to } \$v\}$$

The denominations available are \$1, \$3, and \$8.

Coin Type i	1	2	3
Denomination	1	3	8

Given the completed memoisation table below, determine which coins are used to make \$7 via backtracking.

Value v	0	1	2	3	4	5	6	7	8	9
MinCoins[v]	0	1	2	1	2	3	2	3	1	2
Decision i	-	1	1	2	1	1	2	1	3	1

Solution

Since a decision row was included, it helps us to backtrack the choices made when the memoisation table was filled. It is these optimal choices that will help determine the specific coins needed to create $7. We start by looking at the optimal decision made to create $7.

Value v	0	1	2	3	4	5	6	**7**	8	9
MinCoins[v]	0	1	2	1	2	3	2	**3**	1	2
Decision i	-	1	1	2	1	1	2	**1**	3	1

The table shows that the decision of using a coin of type 1 was part of the optimal substructure. Hence, we can deduce that the $1 coin was used. After identifying what decision was made, we can reduce our problem to it's underlying optimal subproblem. In this case, since we needed a $1 coin, we can now look at repeating our steps at $6 instead.

Value v	0	1	2	3	4	5	**6**	**7**	8	9
MinCoins[v]	0	1	2	1	2	3	**2**	**3**	1	2
Decision i	-	1	1	2	1	1	**2**	**1**	3	1

For $6, the table shows that a decision of taking a coin of type 2 was made, corresponding to the $3 coin. Hence, the coins used so far are $1 and $3. Like the case before, we can now shift our focus to $6 - $3 = $3 as the remaining total we need to make.

Value v	0	1	2	**3**	4	5	**6**	**7**	8	9
MinCoins[v]	0	1	2	**1**	2	3	**2**	**3**	1	2
Decision i	-	1	1	**2**	1	1	**2**	**1**	3	1

For $3, the table shows that a decision of taking a coin of type 2 was made once again. This refers to another $3 coin being used. Adjusting the total will leave us at the base case of $0 where the backtracking process will finish.

Therefore, the optimal coins to create $7 are $1, $3, and $3, using the provided denominations.

Problem 3. Implement the solution to the coin change problem described in the lectures. Your solution should return the number of coins needed, along with how many of each denomination are required.

(a) Use the bottom-up strategy to compute the solutions.

(b) Use the top-down strategy to compute the solutions.

Consult the notes if you are unclear on the difference between the two approaches.