Prepared by Julian García

Lecture 26
Dynamic programming II

FIT 1008&2085
Introduction to Computer Science

MONASH University
Information Technology

Dynamic Programming

Maximum subsequence sum (part I)

Knapsack (part II)

20 kg max

Knapsack

Knapsack

Suppose you are in a treasure cave which contains 6 precious items, with the following weights and monetary value.

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

You want to take as much treasure as you can carry. However, you can only carry up to 20kg. Which items do you take?

What is the maximum value you can take?

Subproblems

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

Item	1	2	3	4
Weight	20kg	10kg	9kg	4kg
Value	$4000	$3500	$1800	$400

Item	1	2
Weight	20kg	10kg
Value	$4000	$3500

Fewer Items

Less capacity

M[i, j] = "Maximum value of the knapsack problem restricted to the first i items and with capacity j"

$0 \le i \le 6$

$0 \le j \le 20$

assume: weights and capacities are integers.

Item	1	2
Weight	20kg	10kg
Value	$4000	$3500

Item	1	2	3	4
Weight	20kg	10kg	9kg	4kg
Value	$4000	$3500	$1800	$400

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0																					
1																					
2																					
3																					
4																					
5																					
6																					

Key insights

- An element can only be included or not included
- Included items leave less capacity available for other items

j

i

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0																					
1																					
2																					
3																					
4																					
5																					
6																					

Want to get to ALL items, ALL values

<u>Fill in</u> MaxValue

Find how subproblems are related

$M[\boxed{0}, j] = \boxed{0}$

No items, no value no matter how large the knapsack

Initial conditions

$M[0, j] = 0$

No items, no value no matter how large the knapsack

i \ j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1																					
2																					
3																					
4																					
5																					
6																					

M[c, n]

Find how subproblems are related

$M[\boxed{0}, j] = \boxed{0}$

No items, no value no matter how large the knapsack

$M[i, \boxed{0}] = \boxed{0}$

No knapsack, no value no matter how many items

Initial conditions

Base case

MV[0, j] = 0

MV[i, 0] = 0

j →

i ↓	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0																				
2	0																				
3	0																				
4	0																				
5	0																				
6	0																				

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j →

i ↓	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0																				
2	0																				
3	0																				
4	0																				
5	0																				
6	0																				

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0																				
2	0																				
3	0																				
4	0																				
5	0																				
6	0																				

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0																				
3	0																				
4	0																				
5	0																				
6	0																				

20 kg doesn't fit in less than 20kg bag

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0																				
3	0																				
4	0																				
5	0																				
6	0																				

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0																				
3	0																				
4	0																				
5	0																				
6	0																				

But it does fit in a 20kg bag

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j →

i ↓	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0																				
3	0																				
4	0																				
5	0																				
6	0																				

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0																				
3	0																				
4	0																				
5	0																				
6	0																				

Adding item 2 as a possibility makes no difference when j (capacity) < 10

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0											
3	0																				
4	0																				
5	0																				
6	0																				

Adding item 2 as a possibility makes no difference when j (capacity) < 10

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0											
3	0																				
4	0																				
5	0																				
6	0																				

Do not put item 2 in the knapsack: 0 Put item 2 in the knapsack: 3500 + 0

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

i \ j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500										
3	0																				
4	0																				
5	0																				
6	0																				

If item 2 included, then there's 10kg less available for prior items

Do not put item 2 in the knapsack: 0 **Put item 2 in the knapsack: 3500 + 0**

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500										
3	0																				
4	0																				
5	0																				
6	0																				

Do not put item 2 in the knapsack: 0

Put item 2 in the knapsack: 3500 + 0

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500									
3	0																				
4	0																				
5	0																				
6	0																				

Do not put item 2 in the knapsack: 0

Put item 2 in the knapsack: 3500 + 0

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j →

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500									
3	0																				
4	0																				
5	0																				
6	0																				

Do not put item 2 in the knapsack: 0

Put item 2 in the knapsack: 3500 + 0

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500									
3	0																				
4	0																				
5	0																				
6	0																				

Do not put item 2 in the knapsack: 0

Put item 2 in the knapsack: 3500 + 0

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500								
3	0																				
4	0																				
5	0																				
6	0																				

Do not put item 2 in the knapsack: 0

Put item 2 in the knapsack: 3500 + 0

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

i \ j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	
3	0																				
4	0																				
5	0																				
6	0																				

Do not put item 2 in the knapsack: 0 Put item 2 in the knapsack: 3500 + 0

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

i \ j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	
3	0																				
4	0																				
5	0																				
6	0																				

4000 > 3500

Do not put item 2 in the knapsack: 4000

Put item 2 in the knapsack: 3500 + 0

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j →

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0																				
4	0																				
5	0																				
6	0																				

Do not put item 2 in the knapsack: 4000 **Put item 2 in the knapsack: 3500 + 0**

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0																				
4	0																				
5	0																				
6	0																				

Adding item 3 as a possibility makes no difference when j < 9

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0												
4	0																				
5	0																				
6	0																				

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0												
4	0																				
5	0																				
6	0																				

Do not put item 3 in the knapsack: 0 Put item 3 in the knapsack: 1800 + 0

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800											
4	0																				
5	0																				
6	0																				

Do not put item 3 in the knapsack: 0

Put item 3 in the knapsack: 1800 + 0

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800											
4	0																				
5	0																				
6	0																				

3500 > 1800

Do not put item 3 in the knapsack: 3500

Put item 3 in the knapsack: 1800 + 0

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j →

i ↓	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500										
4	0																				
5	0																				
6	0																				

Do not put item 3 in the knapsack: 3500 **Put item 3 in the knapsack: 1800 + 0**

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j →

i ↓	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500									
4	0																				
5	0																				
6	0																				

Do not put item 3 in the knapsack: 3500 Put item 3 in the knapsack: 1800 + 0

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

i \ j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500								
4	0																				
5	0																				
6	0																				

Do not put item 3 in the knapsack: 3500 **Put item 3 in the knapsack: 1800 + 0**

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j →

i ↓	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500		
4	0																				
5	0																				
6	0																				

Do not put item 3 in the knapsack: 3500 **Put item 3 in the knapsack: 1800 + 0**

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500		
4	0																				
5	0																				
6	0																				

3500 < 3500+1800

Do not put item 3 in the knapsack: 3500

Put item 3 in the knapsack: 1800 + 3500

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j →

i ↓	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	
4	0																				
5	0																				
6	0																				

Do not put item 3 in the knapsack: 3500 Put item 3 in the knapsack: 1800 + 3500

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	
4	0																				
5	0																				
6	0																				

4000 < 5300

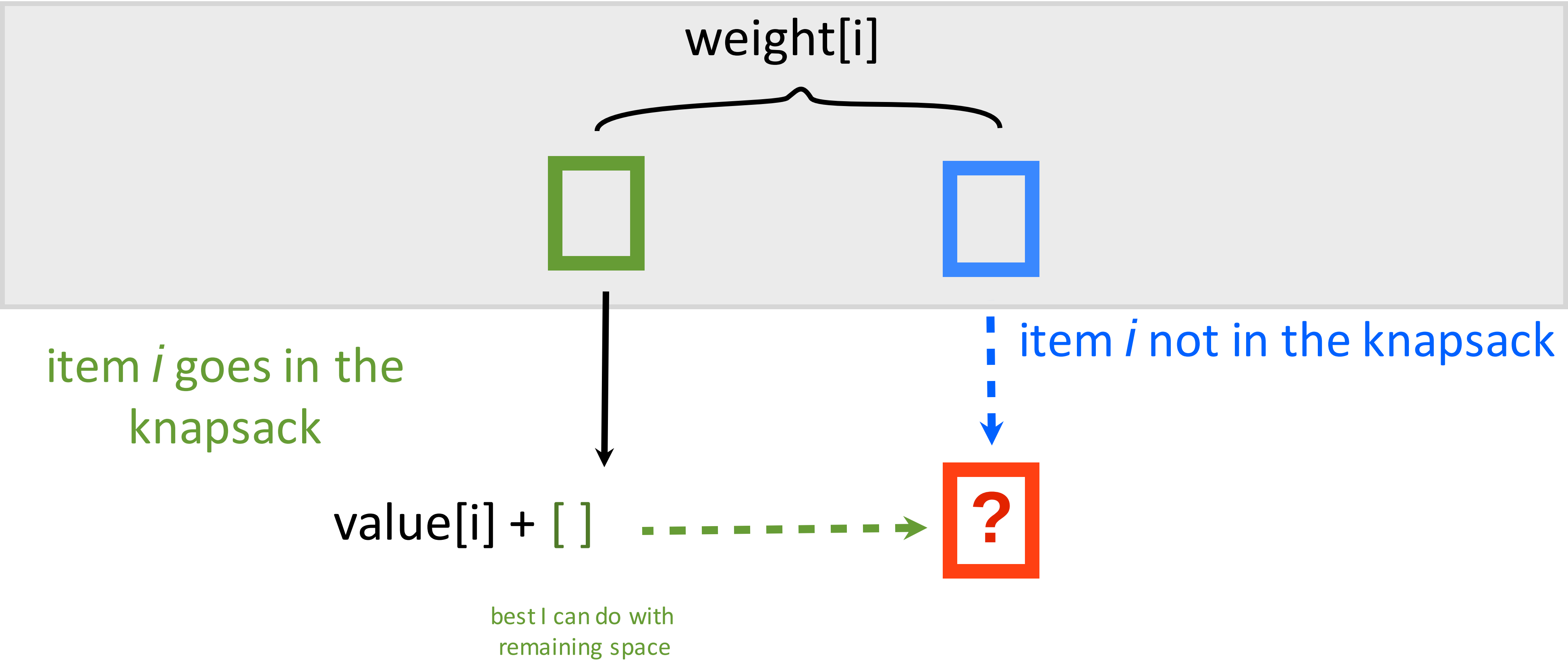
Do not put item 3 in the knapsack: 4000

Put item 3 in the knapsack: 1800 + 3500

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j →

i ↓	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0																				
5	0																				
6	0																				


weight[i]
item i goes in the knapsack
item i not in the knapsack
value[i] + [ ]
?
best I can do with remaining space

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j →

i ↓	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0																				
5	0																				
6	0																				

Adding item 3 as a possibility makes no difference when j < 4

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

i \ j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0																	
5	0																				
6	0																				

Adding item 3 as a possibility makes no difference when j < 4

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i

i \ j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0																	
5	0																				
6	0																				

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0																	
5	0																				
6	0																				

Do not put item 4 in the knapsack: 0

Put item 4 in the knapsack: 400 + 0

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400																
5	0																				
6	0																				

Do not put item 4 in the knapsack: 0

Put item 4 in the knapsack: 400 + 0

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400												
5	0																				
6	0																				

Do not put item 4 in the knapsack: 0 **Put item 4 in the knapsack: 400 + 0**

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400												
5	0																				
6	0																				

Do not put item 4 in the knapsack: 1800 Put item 4 in the knapsack: 400 + 0

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400	1800											
5	0																				
6	0																				

Do not put item 4 in the knapsack: 1800 **Put item 4 in the knapsack: 400 + 0**

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

i \ j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400	1800											
5	0																				
6	0																				

Do not put item 4 in the knapsack: 3500 **Put item 4 in the knapsack: 400 + 0**

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400	1800	3500										
5	0																				
6	0																				

Do not put item 4 in the knapsack: 3500 **Put item 4 in the knapsack: 400 + 0**

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j →

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400	1800	3500	3500									
5	0																				
6	0																				

Do not put item 4 in the knapsack: 3500 **Put item 4 in the knapsack: 400 + 0**

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400	1800	3500	3500	3500								
5	0																				
6	0																				

Do not put item 4 in the knapsack: 3500 **Put item 4 in the knapsack: 400 + 1800**

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j →

i ↓	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400	1800	3500	3500	3500	3500							
5	0																				
6	0																				

Do not put item 4 in the knapsack: 3500 **Put item 4 in the knapsack: 400 + 1800**

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j →

i ↓	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400	1800	3500	3500	3500	3500							
5	0																				
6	0																				

Do not put item 4 in the knapsack: 3500 **Put item 4 in the knapsack: 400 + 3500**

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400	1800	3500	3500	3500	3500	3900						
5	0																				
6	0																				

Do not put item 4 in the knapsack: 3500 **Put item 4 in the knapsack: 400 + 3500**

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400	1800	3500	3500	3500	3500	3900	3900					
5	0																				
6	0																				

Do not put item 4 in the knapsack: 3500

Put item 4 in the knapsack: 400 + 3500

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400	1800	3500	3500	3500	3500	3900	3900	3900	3900	3900		
5	0																				
6	0																				

Do not put item 4 in the knapsack: 3500 **Put item 4 in the knapsack: 400 + 3500**

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i

i \ j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400	1800	3500	3500	3500	3500	3900	3900	3900	3900	3900		
5	0																				
6	0																				

Do not put item 4 in the knapsack: 5300 **Put item 4 in the knapsack: 400 + 3500**

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400	1800	3500	3500	3500	3500	3900	3900	3900	3900	3900	5300	
5	0																				
6	0																				

Do not put item 4 in the knapsack: 5300 **Put item 4 in the knapsack: 400 + 3500**

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j →

i ↓	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400	1800	3500	3500	3500	3500	3900	3900	3900	3900	3900	5300	5300
5	0																				
6	0																				

Do not put item 4 in the knapsack: 5300 **Put item 4 in the knapsack: 400 + 3500**

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400	1800	3500	3500	3500	3500	3900	3900	3900	3900	3900	5300	5300
5	0	0	1000	1000	1000	1000	1400	1400	1400	1800	3500	3500	4500	4500	4500	4500	4900	4900	4900	5300	5300
6	0	200	1000	1200	1200	1200	1400	1600	1600	1800	3500	3700	4500	4700	4700	4700	4900	5100	5100	5300	$\boxed{\textbf{5500}}$

Knapsack

Item *i* does not fit

M[0, j] = 0

M[i, 0] = 0

base case

$$M(i, j) = \begin{cases} M[i-1, j] & \textbf{if } i.weight > j \\ \textbf{Max } \{ M[i-1, j] \textbf{,} \; i.value + M[i-1, j - i.weight] \} & \textbf{otherwise} \end{cases}$$

items

capacity

item *i* not in the knapsack

item *i* goes in the knapsack

j

i

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0																					
1																					
2																					
3																					
4																					
5																					
6																					

Our goal is to find a solution for capacity 20 and items 0-6

Item	1	2
Weight	20kg	10kg
Value	$4000	$3500

10 Kg

j →

i ↓	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0																					
1																					
2																					
3																					
4																					
5																					
6																					

This includes the subproblem of items 0-2 and capacity 10kg

The last item is worth $400, and it weights 4kg. The value in the square marked **?** is:

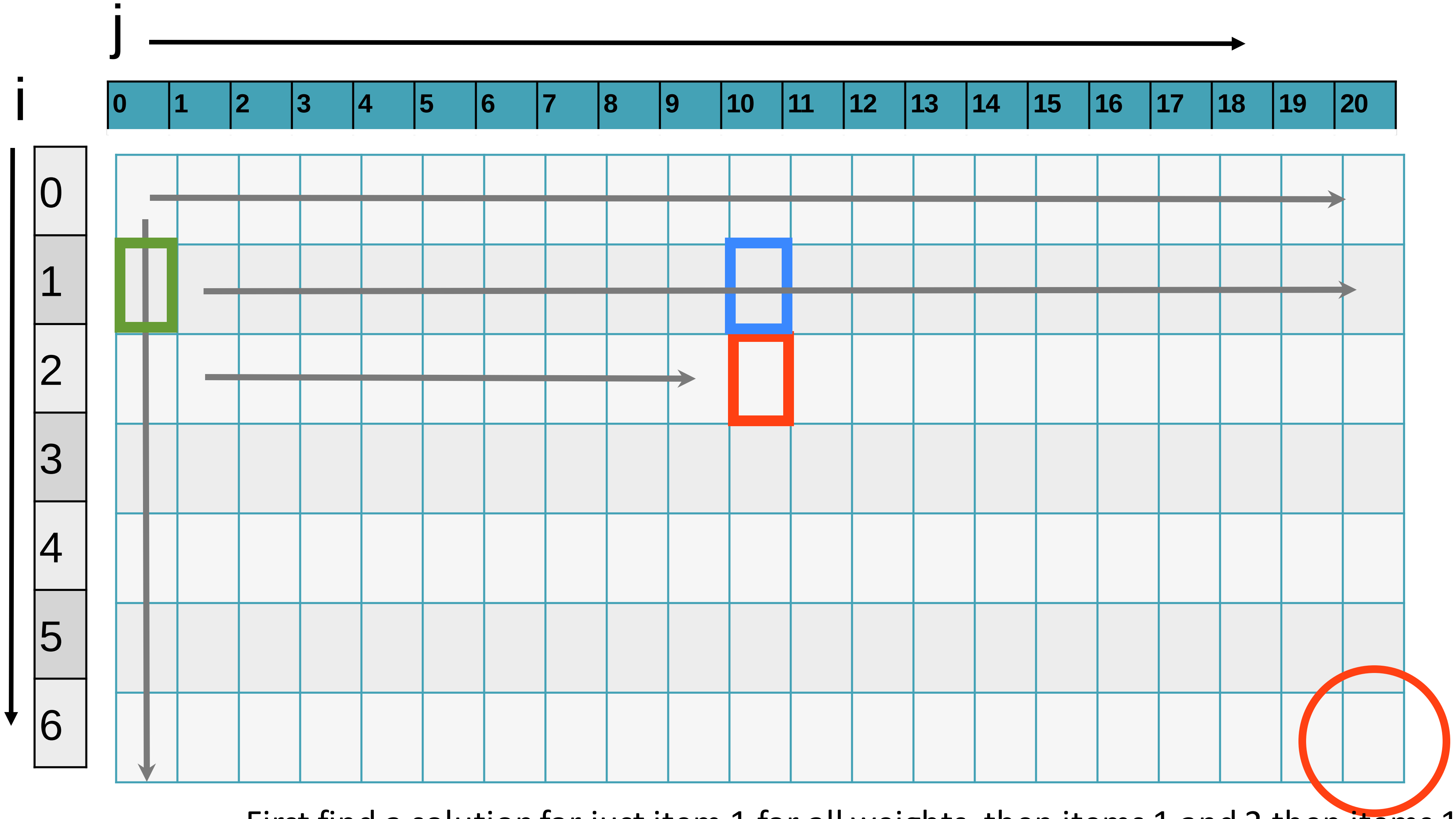
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400	1800	3500	3500	3500	?							

A) 2200

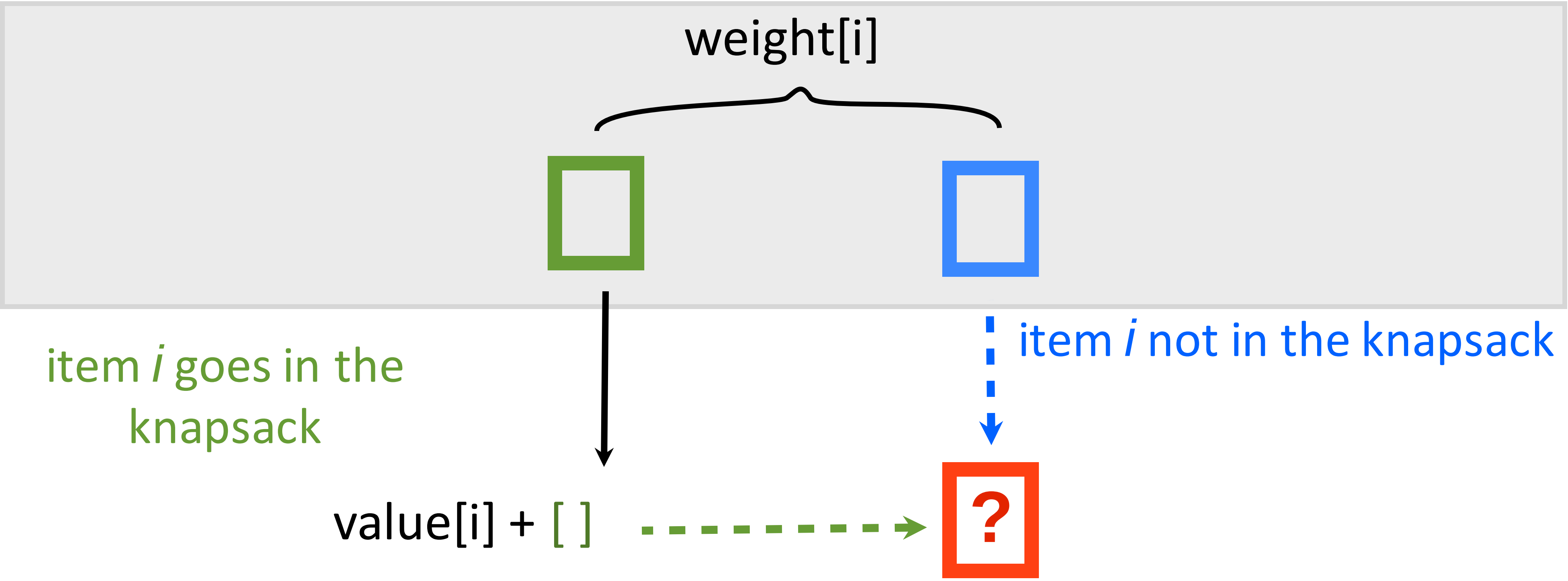
B) 3500

C) 3900

D) None of the above



First find a solution for just item 1 for all weights, then items 1 and 2 then items 1-3, etc.


weight[i]
item i goes in the knapsack
item i not in the knapsack
value[i] + [ ]
?

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400	1800	3500	3500	3500	3500	3900	3900	3900	3900	3900	5300	5300
5	0	0	1000	1000	1000	1000	1400	1400	1400	1800	3500	3500	4500	4500	4500	4500	4900	4900	4900	5300	5300
6	0	200	1000	1200	1200	1200	1400	1600	1600	1800	3500	3700	4500	4700	4700	4700	4900	5100	5100	5300	**5500**

MV[0, j] = 0

base case

MV[i, 0] = 0

$$MV(i, j) = \begin{cases} MV(i-1, j) & \textbf{if } \text{weights}[i] > j \\ \textbf{Max}\ \{\ MV(i-1, j)\ ,\ \text{values}[i] + MV(i-1, j - \text{weights}[i])\ \} & \end{cases}$$

items **capacity**

Knapsack

M[0, j] = 0

M[i, 0] = 0

base case

Item *i* does not fit

M(i, j) =

- M[i-1, j] **if** i.weight > j
- **Max {** M[i-1, j] **,** i.value + M[i-1, j - i.weight] **}** **otherwise**

items **capacity**

item *i* not in the knapsack

item *i* goes in the knapsack

A matrix object using **numpy**

```
import numpy as np
```

```
np.zeros((4, 6))
```

```
array([[ 0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.]])
```

```
my_table = np.zeros((4, 6))
my_table[2, 3] = -1
my_table
```

```
array([[ 0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0., -1.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.,  0.]])
```

```
my_table.shape
```

```
(4, 6)
```

```
(n, m) = my_table.shape
print(n)
print(m)
```

```
4
6
```

enumerate

```
a_list_of_things = ['A', 'B', 'C', 'D', 'E']
```

```
for i, thing in enumerate(a_list_of_things):
    print(i, end=", ")
    print(thing)
```

```
0, A
1, B
2, C
3, D
4, E
```

```
for i, thing in enumerate(a_list_of_things, start=1):
    print(i, end=", ")
    print(thing)
```

```
1, A
2, B
3, C
4, D
5, E
```

```python
class Item:

    def __init__(self, value=0, weight=0):
        assert type(weight) == int
        self.value = value
        self.weight = weight

    def __str__(self):
        # str is meant to be readable, possibly ambiguous
        return "(v={}: w={})".format(self.value, self.weight)

    def __repr__(self):
        # __repr__ is supposed to be unambiguous
        return str(self)
```

The *str* of the standard python list defaults to __*repr*__ for printing its objects

```python
def knapsack_value(list_of_items, knapsack_capacity):
```

```python
def knapsack_value(list_of_items, knapsack_capacity):
    assert len(list_of_items) > 0, "No items"
    assert knapsack_capacity > 0, "No space to store anything?"
    assert type(knapsack_capacity) == int
```

```python
def knapsack_value(list_of_items, knapsack_capacity):
    assert len(list_of_items) > 0, "No items"
    assert knapsack_capacity > 0, "No space to store anything?"
    assert type(knapsack_capacity) == int

    table = np.zeros(shape=(len(list_of_items) + 1, knapsack_capacity + 1))
```

$MV[0, j] = 0$

$MV[i, 0] = 0$

base case

```python
def knapsack_value(list_of_items, knapsack_capacity):
    assert len(list_of_items) > 0, "No items"
    assert knapsack_capacity > 0, "No space to store anything?"
    assert type(knapsack_capacity) == int

    table = np.zeros(shape=(len(list_of_items) + 1, knapsack_capacity + 1))

    for i, item in enumerate(list_of_items, start=1):
        for j in range(1, knapsack_capacity+1):
```

Leave 0 row and 0 column alone.

```python
def knapsack_value(list_of_items, knapsack_capacity):
    assert len(list_of_items) > 0, "No items"
    assert knapsack_capacity > 0, "No space to store anything?"
    assert type(knapsack_capacity) == int

    table = np.zeros(shape=(len(list_of_items) + 1, knapsack_capacity + 1))

    for i, item in enumerate(list_of_items, start=1):
        for j in range(1, knapsack_capacity+1):
            if item.weight > j:
                table[i, j] = table[i-1, j]
```

```python
def knapsack_value(list_of_items, knapsack_capacity):
    assert len(list_of_items) > 0, "No items"
    assert knapsack_capacity > 0, "No space to store anything?"
    assert type(knapsack_capacity) == int

    table = np.zeros(shape=(len(list_of_items) + 1, knapsack_capacity + 1))

    for i, item in enumerate(list_of_items, start=1):
        for j in range(1, knapsack_capacity+1):
            if item.weight > j:
                table[i, j] = table[i-1, j]
            else:
                table[i, j] = max(table[i-1, j], item.value + table[i-1, j - item.weight])
```

$$M(i, j) = \begin{cases} \boxed{M[i-1, j] \quad \textbf{if } i.weight > j} \\ \textbf{Max } \{ \boxed{M[i-1, j]}, \boxed{i.value + M[i-1, j - i.weight]} \} \quad \textbf{otherwise} \end{cases}$$

items **capacity** item *i* not in the knapsack item *i* goes in the knapsack

```python
def knapsack_value(list_of_items, knapsack_capacity):
    assert len(list_of_items) > 0, "No items"
    assert knapsack_capacity > 0, "No space to store anything?"
    assert type(knapsack_capacity) == int

    table = np.zeros(shape=(len(list_of_items) + 1, knapsack_capacity + 1))

    for i, item in enumerate(list_of_items, start=1):
        for j in range(1, knapsack_capacity+1):
            if item.weight > j:
                table[i, j] = table[i-1, j]
            else:
                table[i, j] = max(table[i-1, j], item.value + table[i-1, j - item.weight])

    return table[-1, -1]
```

Find list of Items

Given a completed DP table

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400	1800	3500	3500	3500	3500	3900	3900	3900	3900	3900	5300	5300
5	0	0	1000	1000	1000	1000	1400	1400	1400	1800	3500	3500	4500	4500	4500	4500	4900	4900	4900	5300	5300
6	0	200	1000	1200	1200	1200	1400	1600	1600	1800	3500	3700	4500	4700	4700	4700	4900	5100	5100	5300	**5500**

Find list of Items

Given a completed DP table

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i

i \ j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400	1800	3500	3500	3500	3500	3900	3900	3900	3900	3900	5300	5300
5	0	0	1000	1000	1000	1000	1400	1400	1400	1800	3500	3500	4500	4500	4500	4500	4900	4900	4900	5300	5300
6	0	200	1000	1200	1200	1200	1400	1600	1600	1800	3500	3700	4500	4700	4700	4700	4900	5100	5100	5300	**5500**

Changed
So item 6 used

Weight -= 1

Find list of Items

Given a completed DP table

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400	1800	3500	3500	3500	3500	3900	3900	3900	3900	3900	5300	5300
5	0	0	1000	1000	1000	1000	1400	1400	1400	1800	3500	3500	4500	4500	4500	4500	4900	4900	4900	5300	5300
6	0	200	1000	1200	1200	1200	1400	1600	1600	1800	3500	3700	4500	4700	4700	4700	4900	5100	5100	5300	**5500**

Changed
So item 3 used

Weight -= 9

Find list of Items

Given a completed DP table

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

j

i

i \ j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400	1800	3500	3500	3500	3500	3900	3900	3900	3900	3900	5300	5300
5	0	0	1000	1000	1000	1000	1400	1400	1400	1800	3500	3500	4500	4500	4500	4500	4900	4900	4900	5300	5300
6	0	200	1000	1200	1200	1200	1400	1600	1600	1800	3500	3700	4500	4700	4700	4700	4900	5100	5100	5300	5500

Weight -= 10

Changed
So item 2 used

Find list of Items

Given a completed DP table

Item	1	2	3	4	5	6
Weight	20kg	10kg	9kg	4kg	2kg	1kg
Value	$4000	$3500	$1800	$400	$1000	$200

No room to the left, no further items used (weight remaining = 0)

i \ j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
2	0	0	0	0	0	0	0	0	0	0	3500	3500	3500	3500	3500	3500	3500	3500	3500	3500	4000
3	0	0	0	0	0	0	0	0	0	1800	3500	3500	3500	3500	3500	3500	3500	3500	3500	5300	5300
4	0	0	0	0	400	400	400	400	400	1800	3500	3500	3500	3500	3900	3900	3900	3900	3900	5300	5300
5	0	0	1000	1000	1000	1000	1400	1400	1400	1800	3500	3500	4500	4500	4500	4500	4900	4900	4900	5300	5300
6	0	200	1000	1200	1200	1200	1400	1600	1600	1800	3500	3700	4500	4700	4700	4700	4900	5100	5100	5300	**5500**

Complexity

- If n is the number of items, and w is the capacity of the Knapsack we have ***O(nw)*** operations.

- A brute force approach (not the algorithm we presented) would be ***$O(2^n)$*** — consider all subsets.

- Note that for the complexity **w** refers to *numeric value of the input, not the number of inputs.* As a function of the number of bits required to represent w, the algorithm is **not polynomial.** This is known as a **pseudo-polynomial algorithm**. Not important for this course… will be discussed in more detail in other units.

Questions

Can you add items whose weight is less than 1 kg?

Can you take fractions of items using this technique?

Can you add items whose weight is less than 1 kg?

A) Yes, modifying the algorithm

B) Yes, as is

C) No, not possible with this technique

Can you take fractions of items using this technique? (If this led to a higher value)

A) Yes, modifying the algorithm

B) Yes, as is

C) No, not possible with this technique

Dynamic Programming

- Optimisation problems.
- Solves problems by **solving subproblems**.
- Subproblems may **overlap**.
- Each subproblem is solved only once, storing relevant information.