KNAPSACK PROBLEM

DYNAMIC PROGRAMMING

Knapsack problem

- Its is a problem in combinatorial optimization
- Given a set of items, each with a mass ***w*** and a value ***v***, determine the number of each item to include in a collection so that the total weight ***M*** is less than or equal to a given limit and the total value is as large as possible
- The problem often arises in resource allocation where there are financial constraints

Applications

- Has lots of applications of course
- Finding the least wasteful way to cut raw materials
- Selection of investments and portfolios
- Selection of assets for asset-backed securitization
- Construction and scoring of tests in which the test-takers have a choice as to which questions they answer

Divisible problem

- If we can take fractions of the given items, then the greedy approach can be used
- Sort the items according to their values, it can be done in **O(N*logN)** time complexity
- Start with the item that is the most valuable and take as much as possible
- Than try with the next item from our sorted list
- This linear search has **O(N)** time complexity
- Overall complexity: **O(N*logN) + O(N) = O(N*logN)** !!!
- So we can solve the divisible knapsack problem quite fast

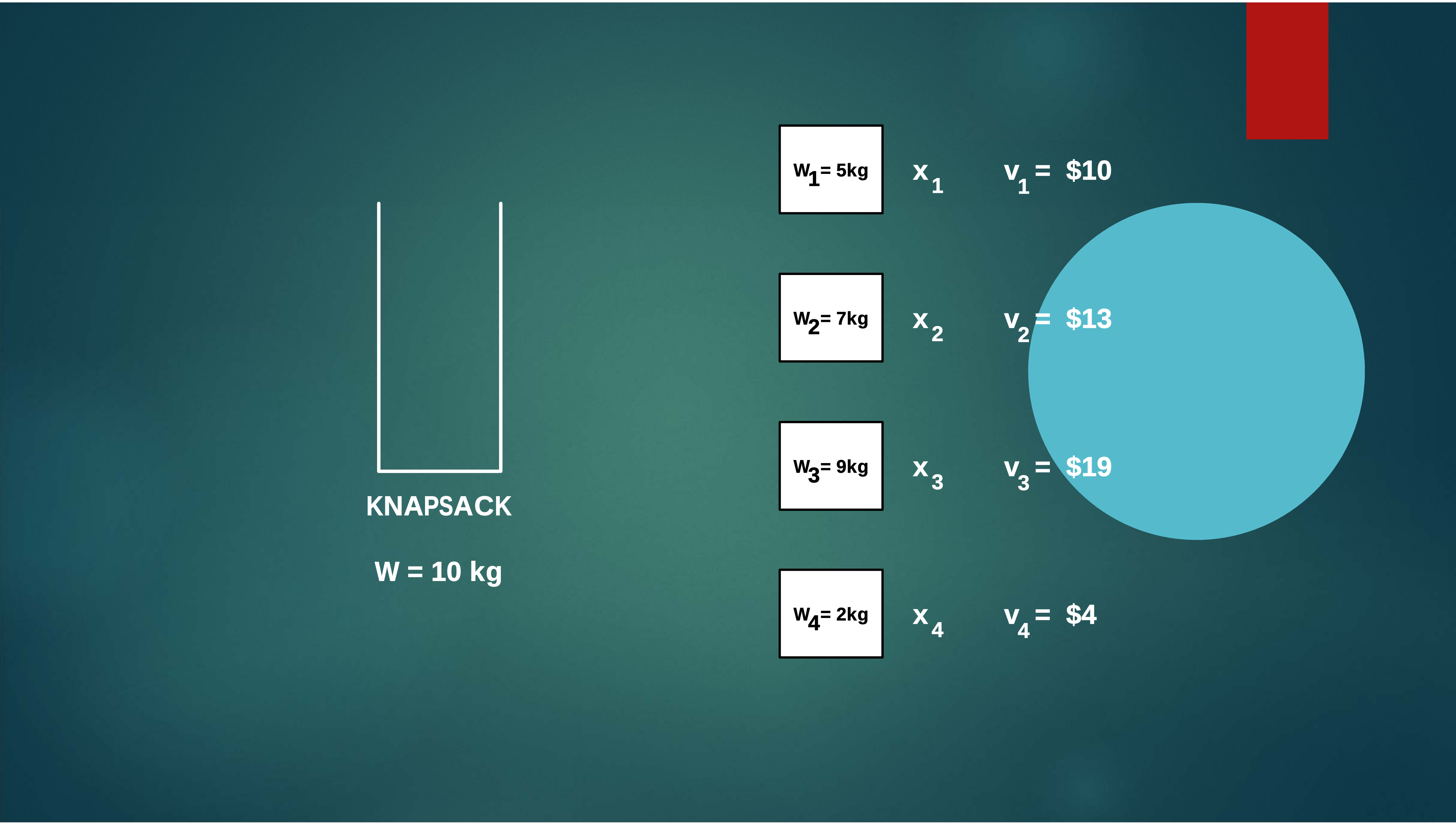
0-1 knapsack problem

- In this case we are not able to take fractions: we have to decide whether to take an item or not
- Greedy algorithm will not provide the optimal result !!!
- Another approach would be to sort by cost per unit weight and include from highest on down until knapsack is full … not a good solution too
- **Dynamic programming** is the right way !!!

Dynamic programming

- Solves larger problem by relating it to overlapping subproblems and then solves the subproblems
- It works through the exponential set of solutions, but does not examine them all explicitly
- Stores intermediate results so that they are not recomputed

  „**memoization**”
- Solution to original problem is easily computed from the solutions to the subproblems


KNAPSACK
W = 10 kg
w_1 = 5kg
x_1
v_1 = \$10
w_2 = 7kg
x_2
v_2 = \$13
w_3 = 9kg
x_3
v_3 = \$19
w_4 = 2kg
x_4
v_4 = \$4

x_i items we have // we have **N** items

v_i value of the i-th item

w_i weight of the i-th item

W maximum capacity of knapsack

---

maximize $\sum_{i=1}^{N} v_i * x_i$ **subject to** $\sum_{i=1}^{N} w_i * x_i \leq W$

x_i **0 if we do not take the i-th item, 1 if we take it**

Knapsack with dynamic programming

- We have to define subproblems: we have **N** items so we have to make **N** decisions whether to take the item with given index or not
- The <u>subproblems</u>: the solution considering every possible combination of remaining items and remaining weight
- **S[i][w]** the solution to the subproblem corresponding to the first **i** items and available weight **w**
- Or in other words...
- **S[i][w]** = the maximum cost of items that fit inside a knapsack of size (weight) **w**, choosing from the first **i** items !!!
- We have to decide whether to take the item or not

S[i][w] = Math.max(S[i-1][w] ; v_i + S[i-1][w-w_i])

do not take i-th item | **we take i-th item**

BUT !!! We are only considering **S[i-1][w-w_i]** if it can fit **$w \geq w_i$**

If there is not room for it → the answer is just **S[i-1][w]** !!!

Time complexity

- Running time of Knapsack: **O(n*W)**
- **BUT** it is not polynomial, it is pseudo-polynomial
- Numeric algorithm runs in **pseudo-polynomial time** if its running time is polynomial in the *numeric value* of the input, but is exponential in the *length* of the input (the number of bits required to represent it)

EXAMPLE

N = 3 items **W = 5kg** capacity of knapsack

Item #1	$w_1 = 4kg$	$v_1 = \$10$
Item #2	$w_2 = 2kg$	$v_2 = \$4$
Item #3	$w_3 = 3kg$	$v_3 = \$7$

		0	1	2	3	4	5
No items	**0**						
First item	**1**						
First two items	**2**						
All items	**3**						

weights [kg]

$$S[i][w] = Math.max(S[i-1][w] \quad ; \quad v_i + S[i-1][w-w_i])$$

N = 3 items   **W = 5kg** capacity of knapsack

Item #1	$w_1 = 4kg$	$v_1 = \$10$
Item #2	$w_2 = 2kg$	$v_2 = \$4$
Item #3	$w_3 = 3kg$	$v_3 = \$7$

		0	1	2	3	4	5
No items	**0**	0	0	0	0	0	0
First item	**1**						
First two items	**2**						
All items	**3**						

weights [kg]

$$S[i][w] = Math.max(S[i-1][w] \; ; \; v_i + S[i-1][w-w_i])$$

N = 3 items **W = 5kg** capacity of knapsack

Item #1 $w_1 = 4kg$ $v_1 = \$10$
Item #2 $w_2 = 2kg$ $v_2 = \$4$
Item #3 $w_3 = 3kg$ $v_3 = \$7$

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0					
First two items	2	0					
All items	3	0					

weights [kg]

$$S[i][w] = Math.max(S[i-1][w] \quad ; \quad v_i + S[i-1][w-w_i])$$

N = 3 items **W = 5kg** capacity of knapsack

Item #1 $\mathbf{w_1}$ = 4kg $\mathbf{v_1}$ = \$10
Item #2 $\mathbf{w_2}$ = 2kg $\mathbf{v_2}$ = \$4
Item #3 $\mathbf{w_3}$ = 3kg $\mathbf{v_3}$ = \$7

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0	0				
First two items	2	0					
All items	3	0					

weights [kg]

S[i][w] = Math.max(S[i-1][w] ; v_i+ S[i-1][w-w_i])

N = 3 items **W = 5kg** capacity of knapsack

Item #1 $w_1 = 4kg$ $v_1 = \$10$

Item #2 $w_2 = 2kg$ $v_2 = \$4$

Item #3 $w_3 = 3kg$ $v_3 = \$7$

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0	0	0			
First two items	2	0					
All items	3	0					

weights [kg]

$$S[i][w] = Math.max(S[i-1][w] \quad ; \quad v_i + S[i-1][w-w_i])$$

N = 3 items **W = 5kg** capacity of knapsack

Item #1	$w_1 = 4kg$	$v_1 = \$10$
Item #2	$w_2 = 2kg$	$v_2 = \$4$
Item #3	$w_3 = 3kg$	$v_3 = \$7$

weights [kg]

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0	0	0	0		
First two items	2	0					
All items	3	0					

$$S[i][w] = Math.max(S[i-1][w] \quad ; \quad v_i + S[i-1][w-w_i])$$

N = 3 items **W = 5kg** capacity of knapsack

Item #1 $w_1 = 4kg$ $v_1 = \$10$

Item #2 $w_2 = 2kg$ $v_2 = \$4$

Item #3 $w_3 = 3kg$ $v_3 = \$7$

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0	0	0	0	10	
First two items	2	0					
All items	3	0					

weights [kg]

$$S[i][w] = Math.max(S[i-1][w] \quad ; \quad v_i + S[i-1][w-w_i])$$

N = 3 items **W = 5kg** capacity of knapsack

Item #1 $w_1 = 4kg$ $v_1 = \$10$

Item #2 $w_2 = 2kg$ $v_2 = \$4$

Item #3 $w_3 = 3kg$ $v_3 = \$7$

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0	0	0	0	10	10
First two items	2	0					
All items	3	0					

weights [kg]

$$S[i][w] = Math.max(S[i-1][w] \quad ; \quad v_i + S[i-1][w-w_i])$$

N = 3 items **W = 5kg** capacity of knapsack

Item #1 $w_1 = 4kg$ $v_1 = \$10$

Item #2 $w_2 = 2kg$ $v_2 = \$4$

Item #3 $w_3 = 3kg$ $v_3 = \$7$

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0	0	0	0	10	10
First two items	2	0	0				
All items	3	0					

weights [kg]

$$S[i][w] = Math.max(S[i-1][w] \quad ; \quad v_i + S[i-1][w-w_i])$$

$$S[2][1] = Math.max(S[1][1] ; \$4 + S[1,1-2]) = max(0,0)$$

N = 3 items **W = 5kg** capacity of knapsack

Item #1	$w_1 = 4kg$	$v_1 = \$10$
Item #2	$w_2 = 2kg$	$v_2 = \$4$
Item #3	$w_3 = 3kg$	$v_3 = \$7$

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0	0	0	0	10	10
First two items	2	0	0	4			
All items	3	0					

weights [kg]

$$S[i][w] = Math.max(\ S[i-1][w]\ ;\ v_i + S[i-1][w-w_i]\ )$$

$$S[2][2] = Math.max(\ S[1][2]\ ;\ \$4 + S[1,2-2]\ ) = max(0,4)$$

N = 3 items **W = 5kg** capacity of knapsack

Item #1	$w_1 = 4kg$	$v_1 = \$10$
Item #2	$w_2 = 2kg$	$v_2 = \$4$
Item #3	$w_3 = 3kg$	$v_3 = \$7$

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0	0	0	0	10	10
First two items	2	0	0	4	4		
All items	3	0					

weights [kg]

$$S[i][w] = Math.max(S[i-1][w] \quad ; \quad v_i + S[i-1][w-w_i])$$

$$S[2][3] = Math.max(S[1][3] ; \$4 + S[1,3-2]) = max(0,4)$$

N = 3 items **W = 5kg** capacity of knapsack

Item #1	$w_1 = 4kg$	$v_1 = \$10$
Item #2	$w_2 = 2kg$	$v_2 = \$4$
Item #3	$w_3 = 3kg$	$v_3 = \$7$

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0	0	0	0	10	10
First two items	2	0	0	4	4	10	
All items	3	0					

weights [kg]

$$S[i][w] = Math.max(\; S[i-1][w] \; ; \; v_i + S[i-1][w-w_i] \;)$$

$$S[2][4] = Math.max(\; S[1][4] \; ; \; \$4 + S[1,4-2] \;) = max(10,4)$$

N = 3 items **W = 5kg** capacity of knapsack

Item #1 $w_1 = 4kg$ $v_1 = \$10$

Item #2 $w_2 = 2kg$ $v_2 = \$4$

Item #3 $w_3 = 3kg$ $v_3 = \$7$

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0	0	0	0	10	10
First two items	2	0	0	4	4	10	10
All items	3	0					

weights [kg]

S[i][w] = Math.max(S[i-1][w] ; v_i + S[i-1][w-w_i])

S[2][5] = Math.max(S[1][5] ; $4 + S[1,5-2]) = max(10,4)

N = 3 items **W = 5kg** capacity of knapsack

Item #1 $w_1 = 4kg$ $v_1 = \$10$

Item #2 $w_2 = 2kg$ $v_2 = \$4$

Item #3 $w_3 = 3kg$ $v_3 = \$7$

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0	0	0	0	10	10
First two items	2	0	0	4	4	10	10
All items	3	0	0				

weights [kg]

S[i][w] = Math.max(S[i-1][w] ; v_i+ S[i-1][w-w_i])

S[3][1] = Math.max(S[2][1] ; $7 + S[2,1-3]) = max(0,0)

N = 3 items **W = 5kg** capacity of knapsack

Item #1 $w_1 = 4kg$ $v_1 = \$10$

Item #2 $w_2 = 2kg$ $v_2 = \$4$

Item #3 $w_3 = 3kg$ $v_3 = \$7$

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0	0	0	0	10	10
First two items	2	0	0	4	4	10	10
All items	3	0	0	4			

weights [kg]

S[i][w] = Math.max(S[i-1][w] ; v_i + S[i-1][w-w_i])

S[3][2] = Math.max(S[2][2] ; \$7 + S[2,2-3]) = max(4,0)

N = 3 items **W = 5kg** capacity of knapsack

Item #1 $\mathbf{w_1} = 4kg$ $\mathbf{v_1} = \$10$

Item #2 $\mathbf{w_2} = 2kg$ $\mathbf{v_2} = \$4$

Item #3 $\mathbf{w_3} = 3kg$ $\mathbf{v_3} = \$7$

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0	0	0	0	10	10
First two items	2	0	0	4	4	10	10
All items	3	0	0	4	7		

weights [kg]

$$S[i][w] = Math.max(S[i-1][w] \; ; \; v_i + S[i-1][w-w_i])$$

$$S[3][3] = Math.max(S[2][3] \; ; \; \$7 + S[2,3-3]) = max(4,7)$$

N = 3 items **W = 5kg** capacity of knapsack

Item #1 $w_1 = 4kg$ $v_1 = \$10$

Item #2 $w_2 = 2kg$ $v_2 = \$4$

Item #3 $w_3 = 3kg$ $v_3 = \$7$

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0	0	0	0	10	10
First two items	2	0	0	4	4	10	10
All items	3	0	0	4	7	10	

weights [kg]

S[i][w] = Math.max(S[i-1][w] ; v_i+ S[i-1][w-w_i])

S[3][4] = Math.max(S[2][4] ; \$7 + S[2,4-3]) = max(10,7)

N = 3 items **W = 5kg** capacity of knapsack

Item #1	$w_1 = 4kg$	$v_1 = \$10$
Item #2	$w_2 = 2kg$	$v_2 = \$4$
Item #3	$w_3 = 3kg$	$v_3 = \$7$

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0	0	0	0	10	10
First two items	2	0	0	4	4	10	10
All items	3	0	0	4	7	10	11

weights [kg]

S[i][w] = Math.max(S[i-1][w] ; v_i+ S[i-1][w-w_i])

S[3][5] = Math.max(S[2][5] ; $7 + S[2,5-3]) = max(10,11)

N = 3 items **W = 5kg** capacity of knapsack

Item #1 $\mathbf{w_1}$ = 4kg $\mathbf{v_1}$ = $10
Item #2 $\mathbf{w_2}$ = 2kg $\mathbf{v_2}$ = $4
Item #3 $\mathbf{w_3}$ = 3kg $\mathbf{v_3}$ = $7

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0	0	0	0	10	10
First two items	2	0	0	4	4	10	10
All items	3	0	0	4	7	10	11

weights [kg]

Solution: value in the last row and last column !!!
→ we can make **$11** from the items we have taken. But what are these items?

N = 3 items **W = 5kg** capacity of knapsack

Item #1 $\mathbf{w_1}$ = 4kg $\mathbf{v_1}$ = $10
Item #2 $\mathbf{w_2}$ = 2kg $\mathbf{v_2}$ = $4
Item #3 $\mathbf{w_3}$ = 3kg $\mathbf{v_3}$ = $7

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0	0	0	0	10	10
First two items	2	0	0	4	4	10	10
All items	3	0	0	4	7	10	11

weights [kg]

Solution: value in the last row and last column !!!
→ we can make **$11** from the items we have taken. But what are these items?

N = 3 items **W = 5kg** capacity of knapsack

Item #1 $\mathbf{w_1}$ = 4kg $\mathbf{v_1}$ = $10

Item #2 $\mathbf{w_2}$ = 2kg $\mathbf{v_2}$ = $4

Item #3 $\mathbf{w_3}$ = 3kg $\mathbf{v_3}$ = $7

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0	0	0	0	10	10
First two items	2	0	0	4	4	10	10
All items	3	0	0	4	7	10	11

weights [kg]

Solution: value in the last row and last column !!!
→ we can make **$11** from the items we have taken. But what are these items?

N = 3 items **W = 5kg** capacity of knapsack

Item #1 $\mathbf{w_1}$ = 4kg $\mathbf{v_1}$ = $10
Item #2 $\mathbf{w_2}$ = 2kg $\mathbf{v_2}$ = $4
Item #3 $\mathbf{w_3}$ = 3kg $\mathbf{v_3}$ = $7

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0	0	0	0	10	10
First two items	2	0	0	4	4	10	10
All items	3	0	0	4	7	10	11

weights [kg]

Solution: value in the last row and last column !!!
→ we can make **$11** from the items we have taken. But what are these items?

N = 3 items **W = 5kg** capacity of knapsack

Item #1 $w_1 = 4kg$ $v_1 = \$10$

Item #2 $w_2 = 2kg$ $v_2 = \$4$

Item #3 $w_3 = 3kg$ $v_3 = \$7$

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0	0	0	0	10	10
First two items	2	0	0	4	4	10	10
All items	3	0	0	4	7	10	11

weights [kg]

Solution: value in the last row and last column !!!

→ we can make **$11** from the items we have taken. But what are these items?

N = 3 items **W = 5kg** capacity of knapsack

Item #1 **w_1** = 4kg **v_1** = \$10
Item #2 **w_2** = 2kg **v_2** = \$4
Item #3 **w_3** = 3kg **v_3** = \$7

		0	1	2	3	4	5
No items	0	0	0	0	0	0	0
First item	1	0	0	0	0	10	10
First two items	2	0	0	4	4	10	10
All items	3	0	0	4	7	10	11

weights [kg]

Solution: value in the last row and last column !!!
→ we can make **\$11** from the items we have taken. But what are these items?