

# Knapsack Problem

Dada una mochila con capacidad  $W$ , y un conjunto de ítems de valor  $v_i$  y peso  $w_i$ ; deseamos obtener los ítems cuyo peso sumado no supere  $W$  y su valor total sea el máximo posible entre todas las posibles combinaciones de ítems.

Este es un problema **np-completo**



Dado el siguiente grupo de ítems, para una mochila cuyo peso máximo  $W$  es igual a 7, resolver el Knapsack Problem



$$W_{\max} = 7$$

**Item 1**

**Value: 16**  
**Weight: 2**

**Item 2**

**Value: 19**  
**Weight: 3**

**Item 3**

**Value: 23**  
**Weight: 4**

**Item 4**

**Value: 28**  
**Weight: 5**

# Utilizando Dynamic Programming

Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0					
1					
2					
3					
4					
5					
6					
7					
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5

# Utilizando Dynamic Programming

Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0				
1	0				
2	0				
3	0				
4	0				
5	0				
6	0				
7	0				
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5



# Utilizando Dynamic Programming

Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0	0			
1	0	0			
2	0				
3	0				
4	0				
5	0				
6	0				
7	0				
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5



# Utilizando Dynamic Programming

Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0	0			
1	0	0			
2	0	16			
3	0	16			
4	0	16			
5	0	16			
6	0	16			
7	0	16			
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5



# Utilizando Dynamic Programming

Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0	0	0		
1	0	0	0		
2	0	16			
3	0	16			
4	0	16			
5	0	16			
6	0	16			
7	0	16			
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5



$2 - 3 = -1 \Rightarrow 0$



# Utilizando Dynamic Programming

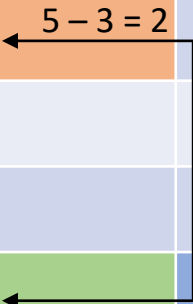
Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0	0	0		
1	0	0	0		
2	0	16	16		
3	0	16			
4	0	16			
5	0	16			
6	0	16			
7	0	16			
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5








# Utilizando Dynamic Programming

Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0	0	0		
1	0	0	0		
2	0	16	16		
3	0	16	19		
4	0	16	19		
5	0	16			
6	0	16			
7	0	16			
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5






# Utilizando Dynamic Programming

Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0	0	0		
1	0	0	0		
2	0	16	16		
3	0	16	19		
4	0	16	19		
5	0	16	35		
6	0	16	35		
7	0	16	35		
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5
					




# Utilizando Dynamic Programming

Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0	0	0	0	
1	0	0	0	0	
2	0	16	16	16	
3	0	16	19	19	
4	0	16	19		
5	0	16	35		
6	0	16	35		
7	0	16	35		
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5



# Utilizando Dynamic Programming




Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0	0	0	0	
1	0	0	0	0	
2	0	16	16	16	
3	0	16	19	19	
4	0	16	19	23	
5	0	16	35	35	
6	0	16	35		
7	0	16	35		
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5



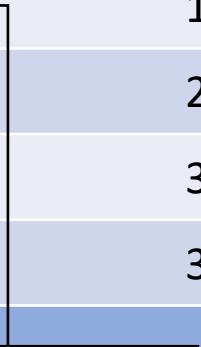
$6 - 4 = 2$

# Utilizando Dynamic Programming

Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0	0	0	0	
1	0	0	0	0	
2	0	16	16	16	
3	0	16	19	19	
4	0	16	19	23	
5	0	16	35	35	
6	0	16	35	39	
7	0	16	35		
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5



$7 - 4 = 3$






# Utilizando Dynamic Programming

Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0	0	0	0	
1	0	0	0	0	
2	0	16	16	16	
3	0	16	19	19	
4	0	16	19	23	
5	0	16	35	35	
6	0	16	35	39	
7	0	16	35	42	
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5



# Utilizando Dynamic Programming



Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	16	16	16	16
3	0	16	19	19	19
4	0	16	19	23	23
5	0	16	35	35	35
6	0	16	35	39	39
7	0	16	35	42	
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5



$7 - 5 = 2$



# Utilizando Dynamic Programming

Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	16	16	16	16
3	0	16	19	19	19
4	0	16	19	23	23
5	0	16	35	35	35
6	0	16	35	39	39
7	0	16	35	42	44
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5
					




# Knapsack Problem




¿Cómo se obtiene la solución al problema. Es decir, los ítems que entraron en la mochila?



# Obtenemos los ítems utilizando el Traceback

Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	16	16	16	16
3	0	16	19	19	19
4	0	16	19	23	23
5	0	16	35	35	35
6	0	16	35	39	39
7	0	16	35	42	44
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5
					

# Obtenemos los ítems utilizando el Traceback

Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	16	16	16	16
3	0	16	19	19	19
4	0	16	19	23	23
5	0	16	35	35	35
6	0	16	35	39	39
7	0	16	35	42	44
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5
					

# Obtenemos los ítems utilizando el Traceback

Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	16	16	16	16
3	0	16	19	19	19
4	0	16	19	23	23
5	0	16	35	35	35
6	0	16	35	39	39
7	0	16	35	42	44
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5





Diagram illustrating the Traceback process for the knapsack problem. The table shows the maximum value for each capacity (0 to 7) and the corresponding items selected. The final value is 44, achieved by selecting Item 3 (Value: 23, Weight: 4) and Item 4 (Value: 28, Weight: 5). The calculation  $44 - 28 = 16$  is shown, indicating the value of the remaining capacity (16) after selecting Item 4.

# Obtenemos los ítems utilizando el Traceback

Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	16	16	16	16
3	0	16	19	19	19
4	0	16	19	23	23
5	0	16	35	35	35
6	0	16	35	39	39
7	0	16	35	42	44
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5









Diagram illustrating the Traceback process for the Knapsack problem. The table shows the maximum value (Value) and weight (Weight) for each capacity (0 to 7) across five items (Item 0 to Item 4). The values are calculated based on the items included in the knapsack. The final value for capacity 7 is 44, which is highlighted in yellow. The diagram shows the path of the optimal solution: Item 0 (Value: 0, Weight: 0) is selected, followed by Item 1 (Value: 16, Weight: 2), then Item 2 (Value: 19, Weight: 3), then Item 3 (Value: 23, Weight: 4), and finally Item 4 (Value: 28, Weight: 5). The text "¿Son iguales?" (Are they equal?) is placed above the value 16 in Item 2, indicating a comparison between the value of Item 2 alone and the value of Item 2 plus Item 1.



# Obtenemos los ítems utilizando el Traceback

Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	16	16	16	16
3	0	16	19	19	19
4	0	16	19	23	23
5	0	16	35	35	35
6	0	16	35	39	39
7	0	16	35	42	44
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5






# Obtenemos los ítems utilizando el Traceback

Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	16	16	16	16
3	0	16	19	19	19
4	0	16	19	23	23
5	0	16	35	35	35
6	0	16	35	39	39
7	0	16	35	42	44
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5



# Obtenemos los ítems utilizando el Traceback




Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	16	16	16	16
3	0	16	19	19	19
4	0	16	19	23	23
5	0	16	35	35	35
6	0	16	35	39	39
7	0	16	35	42	44
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5








# Obtenemos los ítems utilizando el Traceback

Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	16	16	16	16
3	0	16	19	19	19
4	0	16	19	23	23
5	0	16	35	35	35
6	0	16	35	39	39
7	0	16	35	42	44
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5



# Estos son los ítems obtenidos utilizando el Traceback

Capacidad	Item 0	Item 1	Item 2	Item 3	Item 4
0	0	0	0	0	0
1	0	0	0	0	0
2	0	16	16	16	16
3	0	16	19	19	19
4	0	16	19	23	23
5	0	16	35	35	35
6	0	16	35	39	39
7	0	16	35	42	44
	Value: 0 Weight: 0	Value: 16 Weight: 2	Value: 19 Weight: 3	Value: 23 Weight: 4	Value: 28 Weight: 5



La solución del Knapsack Problem para este grupo de ítems, con una mochila de peso máximo  $W$  igual a 7 es tomar el Item 1 y el Item 4



$$W_{\max} = 7$$

**Item 1**

Value: 16  
Weight: 2

**Item 4**

Value: 28  
Weight: 5

$$V_{\text{total}} = 44$$
$$W_{\text{total}} = 7$$

**Item 2**

Value: 19  
Weight: 3

**Item 3**

Value: 23  
Weight: 4