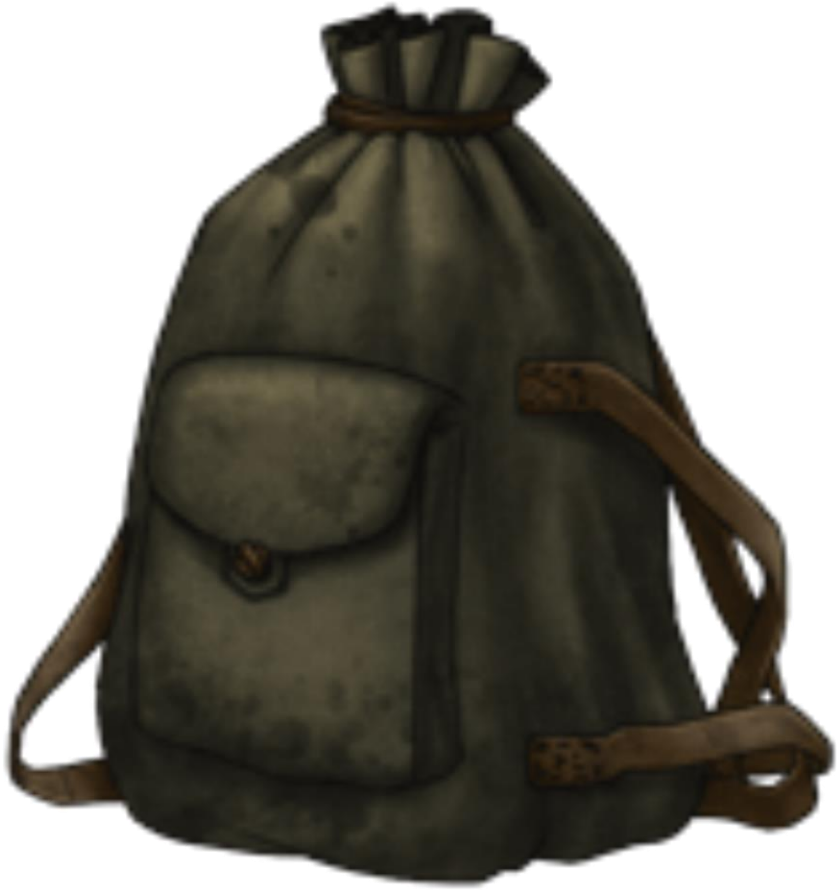
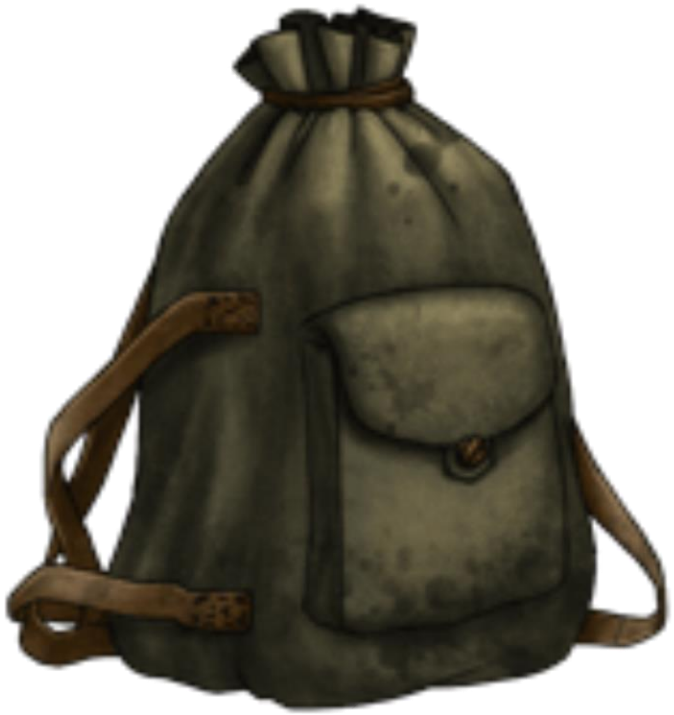
Knapsack Problem

Dada una mochila con capacidad **W**, y un conjunto de ítems de valor $\mathbf{v_i}$ y peso $\mathbf{w_i}$; 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	Item 2	Item 3	Item 4
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					
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

2 – 2 = 0

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 => 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

3 – 3 = 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	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

5 – 3 = 2

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

4 - 4 = 0



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



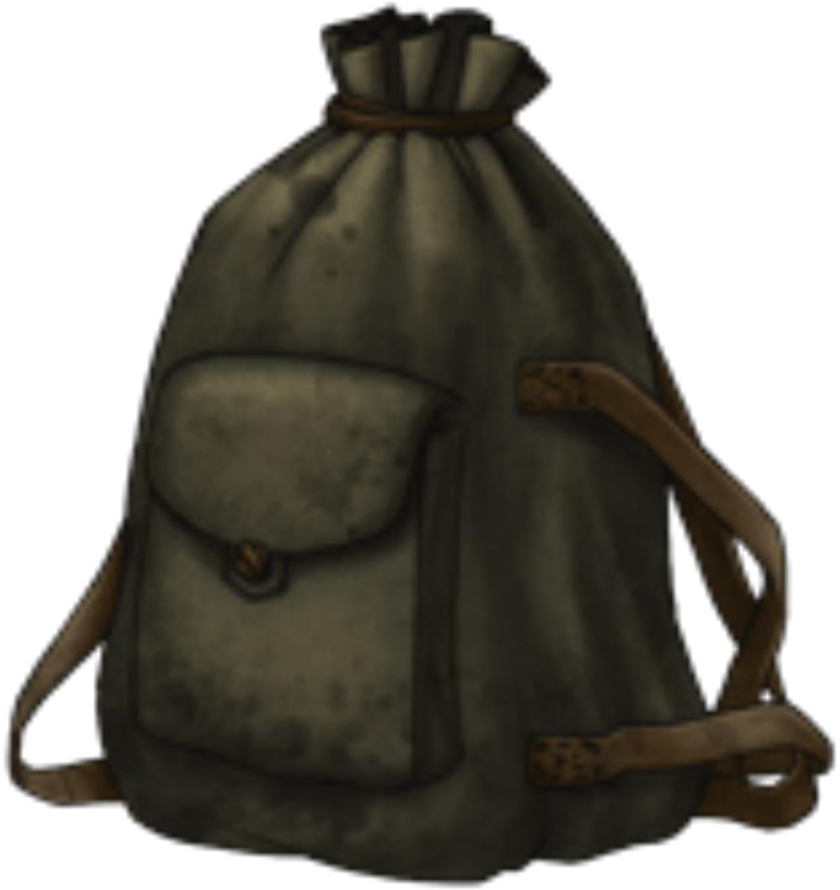
Utilizando Dynamic Programming

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 ¿Son iguales?	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

44 – 28 = 16



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 ¿Son iguales?	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

¿Son iguales?



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

¿Son iguales?



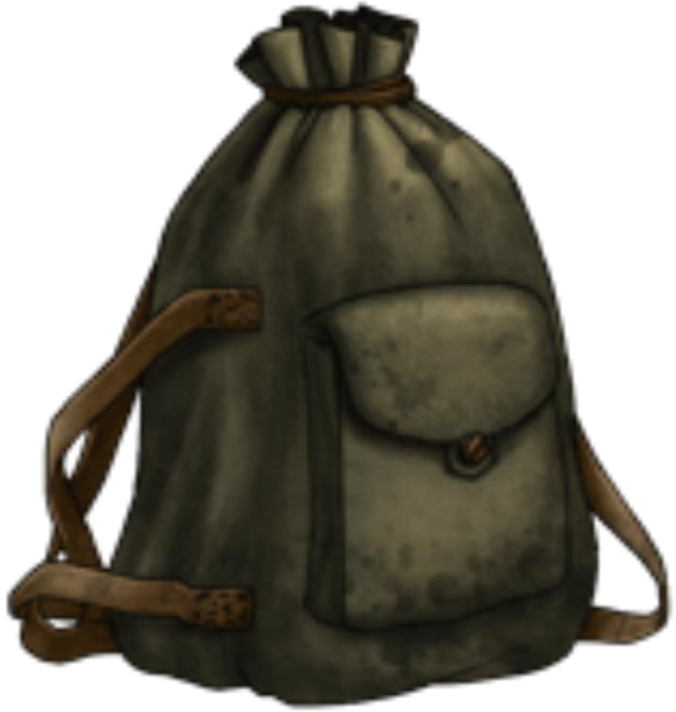
Estos son los ítems obtenidos utilizando el Traceback

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

16 – 16 = 0



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

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

Item 2

Value: 19
Weight: 3

Item 3

Value: 23
Weight: 4