Knapsack Problem

(Dynamic Programming)

Knapsack Problem

Input :- In knapsack problem: There are given n items of known weights w1,...,wn and values v1,...,vn and a knapsack of capacity W .

Output:- Find the most valuable subset of the items that fit into the knapsack with maximum benefit.

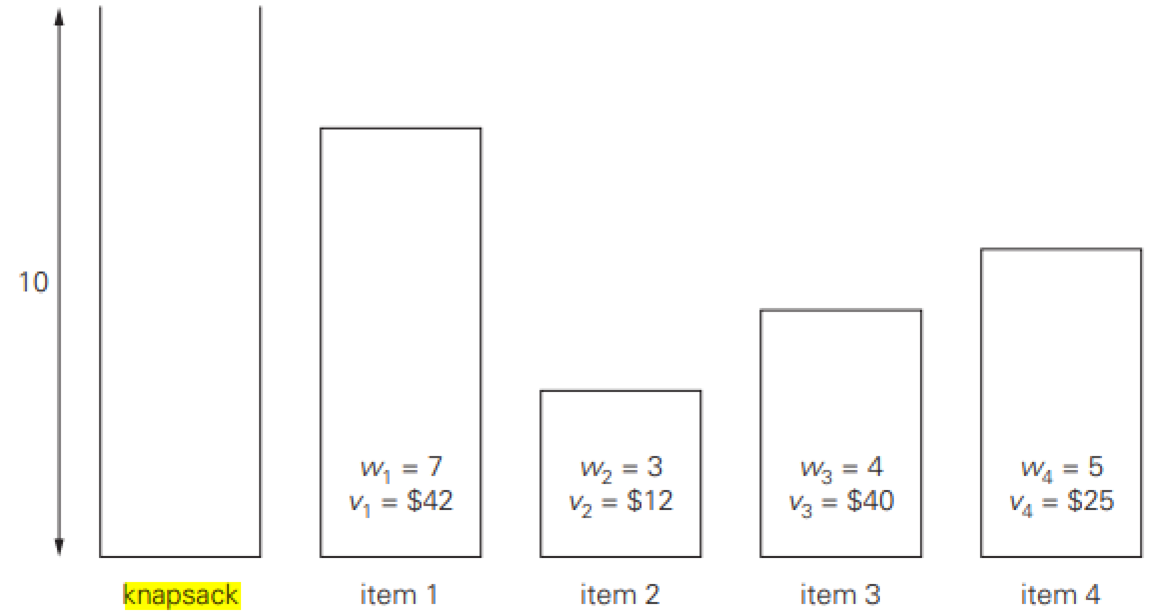
Knapsack problem

EXAMPLE 1 Let us consider the instance given by the following data:

item	weight	value
1	2	\$12
2	1	\$10
3	3	\$20
4	2	\$15

capacity $W = 5$.

Knapsack using Brute Force



Subset	Total weight	Total value
$\varnothing$	0	$ 0
{1}	7	$42
{2}	3	$12
{3}	4	$40
{4}	5	$25
{1, 2}	10	$54
{1, 3}	11	not feasible
{1, 4}	12	not feasible
{2, 3}	7	$52
{2, 4}	8	$37
{3, 4}	**9**	**$65**
{1, 2, 3}	14	not feasible
{1, 2, 4}	15	not feasible
{1, 3, 4}	16	not feasible
{2, 3, 4}	12	not feasible
{1, 2, 3, 4}	19	not feasible

Complexity of Knapsack problem using Brute Force

2^n

Knapsack problem using Dynamic Programming

EXAMPLE 1 Let us consider the instance given by the following data:

item	weight	value
1	2	\$12
2	1	\$10
3	3	\$20
4	2	\$15

capacity $W = 5$.

Formula to solve a problem

$$F(i, j) = \begin{cases} \max\{F(i-1, j),\ v_i + F(i-1, j-w_i)\} & \text{if } j - w_i \ge 0, \\ F(i-1, j) & \text{if } j - w_i < 0. \end{cases}$$

It is convenient to define the initial conditions as follows:

$$F(0, j) = 0 \ \text{ for } j \ge 0 \quad \text{and} \quad F(i, 0) = 0 \ \text{ for } i \ge 0.$$

$$F(i, j) = \begin{cases} \max\{F(i-1, j), v_i + F(i-1, j-w_i)\} & \text{if } j - w_i \geq 0, \\ F(i-1, j) & \text{if } j - w_i < 0. \end{cases}$$

		capacity j					
	i	0	1	2	3	4	5
	0	0	0	0	0	0	0
$w_1 = 2, v_1 = 12$	1	0	0	12	12	12	12
$w_2 = 1, v_2 = 10$	2	0	10	12	22	22	22
$w_3 = 3, v_3 = 20$	3	0	10	12	22	30	32
$w_4 = 2, v_4 = 15$	4	0	10	15	25	30	**37**

Example 2

ITEMS	1	2	3	4
Weights	2	3	4	5
Value	3	7	2	9

0	0	0	0	0	0
0					
0					
0					
0					

Complexity of Knapsack problem using Dynamic Programming

- The time efficiency and space efficiency of this algorithm are both in $O(nW)$.