Design and Analysis of Algorithm (DAA)

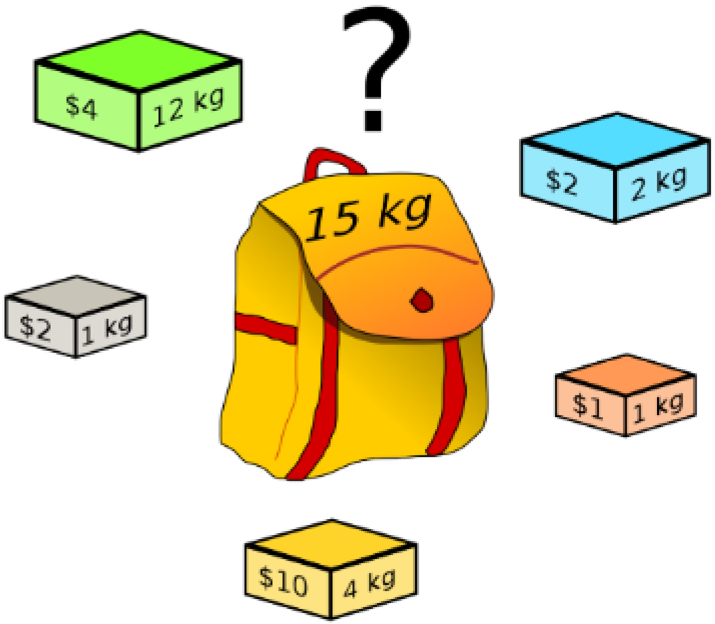
Dynamic Programming

(0/1 Knapsack)

Dr. Dayal Kumar Behera

School of Computer Engineering
KIIT Deemed to be University, Bhubaneswar, India

What is Knapsack Problem?



Knapsack Problem

Select objects to fill the Knapsack such that:

Total weight should not exceed W =15kg(Constraint)

Total Profit should be the maximum

Knapsack Problem

n items or objects

Each item i has:

Weight : w_i

Profit or value : v_i

Knapsack of Capacity: W

Goal: *Find a subset of items with total weight less than or equal to knapsack capacity and total value is maximized.*

Mathematical Interpretation

Objective of the solution:

Maximize : $\sum_{1 \le i \le n} v_i w_i$

Subject to: $\sum_{1 \le i \le n} w_i$ <= W

Types of Knapsack Problem

<u>Knapsack Problem Variants-</u>

Knapsack problem has the following two variants-

1. 0/1 Knapsack Problem: (0-1 decision)

- In this case, either the item is taken completely or left behind. (Fractional amount of an item can not be taken)
- Based on Dynamic Programming

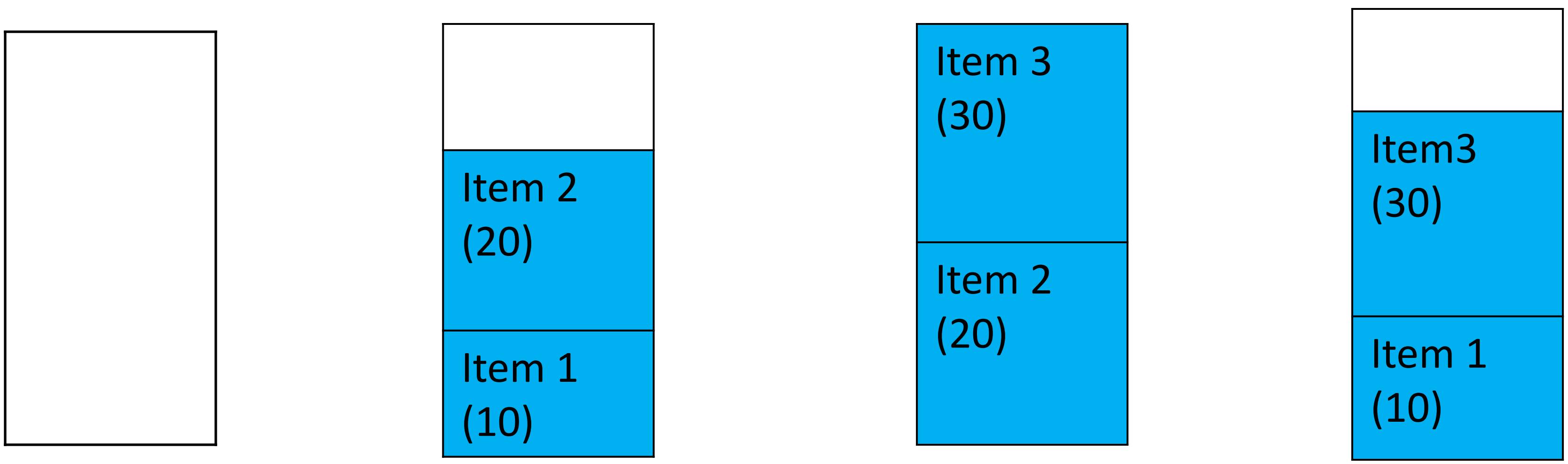
2. Fractional Knapsack Problem:

- In this case, fractional amount of an item can be taken rather than having binary choice.
- Based on Greedy.

Example 1 (0-1 Knapsack)

i	1	2	3
v_i	60	100	120
w_i	10	20	30

Knapsack capacity (W): 50 kg



Total Weight: 0
Profit: 0

Total Weight: 30
Profit: 160

Total Weight: 50
Profit: 220

Total Weight: 40
Profit: 180

0-1 Knapsack: Naïve Approach

i	1	2	3	4
v_i	3	4	5	6
w_i	2	3	4	5

Knapsack capacity (W): 5 kg

Brute-force Approach:

Time complexity: O(2^n)

4	3	2	1	Weight	Profit
0	0	0	0	0	0
0	0	0	1	2	3
0	0	1	0	3	4
…	…	…	…		
1	1	1	1	14 (X)	-

0-1 Knapsack: DP Approach

i	1	2	3	4
v_i	3	4	5	6
w_i	2	3	4	5

Knapsack capacity (W): 5 kg

DP Approach:

Time complexity: O(n * W)

n – No of items
W – Knapsack Capacity

	w	1	2	3	4	5
i	0	0	0	0	0	0
1	0					
2	0					
3	0					
4	0					

0-1 Knapsack: DP Approach

G[i][w]: represent the maximum value that can be obtained with the first **i** items and capacity **w**

The recursive relation is:

$$G[i, w] = \begin{cases} 0 & \text{if } i = 0 \text{ or } w = 0 \\ G[i-1, w] & \text{if } w_i > w \\ \max\left(G[i-1, w],\ v_i + G[i-1, w - w_i]\right) & \text{otherwise} \end{cases}$$

If no items are chosen, the value is zero

If we do not include the ith item, the solution is the same as for the first i-1 items.

If we include the ith item, the gain is the item's value plus the optimal solution for the remaining capacity.

0-1 Knapsack: DP Approach

$$G[i, w] = \begin{cases} 0 & \text{if } i = 0 \text{ or } w = 0 \\ G[i-1, w] & \text{if } w_i > w \\ \max\left(G[i-1, w],\ v_i + G[i-1, w - w_i]\right) & \text{otherwise} \end{cases}$$

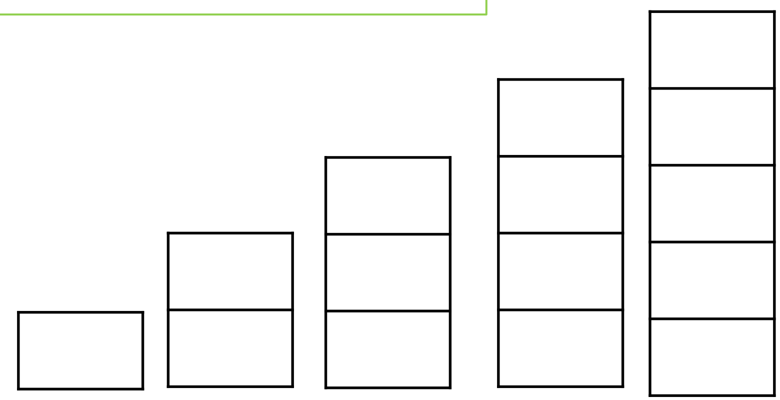
i	1	2	3	4
v_i	3	4	5	6
w_i	2	3	4	5

i \ w	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0					
2	0					
3	0					
4	0					

Knapsack capacity (W): 5 kg

0-1 Knapsack: DP Approach

$$G[i, w] = \begin{cases} 0 & \text{if } i = 0 \text{ or } w = 0 \\ G[i-1, w] & \text{if } w_i > w \\ \max\left(G[i-1, w],\ v_i + G[i-1, w - w_i]\right) & \text{otherwise} \end{cases}$$



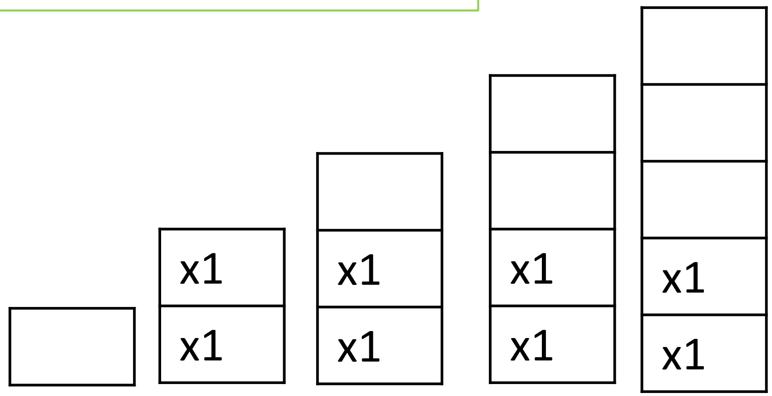
i	1	2	3	4
v_i	3	4	5	6
w_i	2	3	4	5

	w	1	2	3	4	5
i	0	0	0	0	0	0
1	0					
2	0					
3	0					
4	0					

Knapsack capacity (W): 5 kg

0-1 Knapsack: DP Approach

$$G[i,w] = \begin{cases} 0 & \text{if } i = 0 \text{ or } w = 0 \\ G[i-1, w] & \text{if } w_i > w \\ \max\left(G[i-1, w],\ v_i + G[i-1, w - w_i]\right) & \text{otherwise} \end{cases}$$



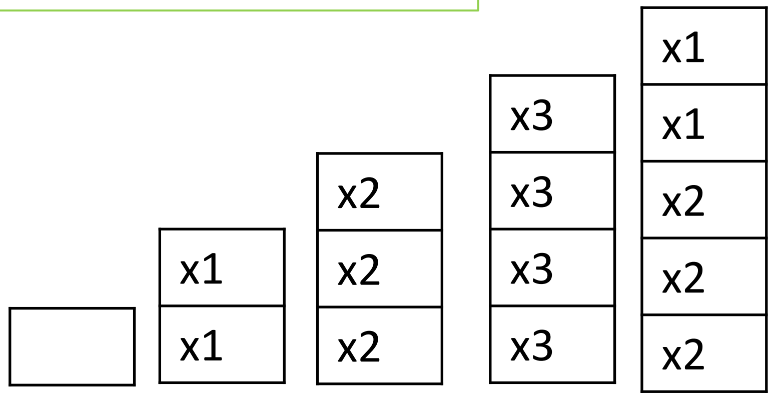
	w	1	2	3	4	5
i	0	0	0	0	0	0
1	0	**0**	**3**	**3**	**3**	**3**
2	0					
3	0					
4	0					

i	**1**	**2**	**3**	**4**
v_i	**3**	**4**	**5**	**6**
w_i	**2**	**3**	**4**	**5**

Knapsack capacity (W): 5 kg

0-1 Knapsack: DP Approach

$$G[i, w] = \begin{cases} 0 & \text{if } i = 0 \text{ or } w = 0 \\ G[i-1, w] & \text{if } w_i > w \\ \max\left(G[i-1, w],\ v_i + G[i-1, w - w_i]\right) & \text{otherwise} \end{cases}$$



i	1	2	3	4
v_i	**3**	**4**	**5**	**6**
w_i	**2**	**3**	**4**	**5**

	w	**1**	**2**	**3**	**4**	**5**
i	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

Knapsack capacity (W): 5 kg

0-1 Knapsack: DP Approach

$$G[i, w] = \begin{cases} 0 & \text{if } i = 0 \text{ or } w = 0 \\ G[i-1, w] & \text{if } w_i > w \\ \max\left(G[i-1, w],\ v_i + G[i-1, w - w_i]\right) & \text{otherwise} \end{cases}$$

1	2	3	4	5
				x1
			x3	x1
		x2	x3	x2
	x1	x2	x3	x2
	x1	x2	x3	x2

i	1	2	3	4
v_i	**3**	**4**	**5**	**6**
w_i	**2**	**3**	**4**	**5**

	w	**1**	**2**	**3**	**4**	**5**
i	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

Knapsack capacity (W): 5 kg

7-4=3

0-1 Knapsack: DP Approach

$$G[i, w] = \begin{cases} 0 & \text{if } i = 0 \text{ or } w = 0 \\ G[i-1, w] & \text{if } w_i > w \\ \max\left(G[i-1, w],\ v_i + G[i-1, w - w_i]\right) & \text{otherwise} \end{cases}$$

1	2	3	4	5
				x1
			x3	x1
		x2	x3	x2
	x1	x2	x3	x2
	x1	x2	x3	x2

i	1	2	3	4
v_i	3	4	5	6
w_i	2	3	4	5

	w	1	2	3	4	5
i	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

Knapsack capacity (W): 5 kg

3-3=0

0-1 Knapsack: DP Approach

```
Algorithm DP_KNAPSACK(v, w, W, n)
for w ← 0 to W do
            G[0, w] = 0
for i ← 1 to n do
            G[i, 0] = 0
for i ← 1 to n do
    for w ← 1 to W do
            if w[i] > w do
                G[i, w] = G[i-1, w]
            else
                G[i, w] =max( G[i-1, w], v[i]+G[i-1,  w-w[i]])
    //end for
//end for
return G[n, W]
```

Example-2

$$G[i, w] = \begin{cases} 0 & \text{if } i = 0 \text{ or } w = 0 \\ G[i-1, w] & \text{if } w_i > w \\ \max\left(G[i-1, w],\ v_i + G[i-1, w - w_i]\right) & \text{otherwise} \end{cases}$$

i	1	2	3	4
v_i	1	2	5	6
w_i	2	3	4	5

Knapsack capacity (W): 8 kg

	w	1	2	3	4	5	6	7	8
i	0	0	0	0	0	0	0	0	0
1	0								
2	0								
3	0								
4	0								

Example-2

$$G[i, w] = \begin{cases} 0 & \text{if } i = 0 \text{ or } w = 0 \\ G[i-1, w] & \text{if } w_i > w \\ \max\left(G[i-1, w],\ v_i + G[i-1, w - w_i]\right) & \text{otherwise} \end{cases}$$

i	1	2	3	4
v_i	1	2	5	6
w_i	2	3	4	5

Knapsack capacity (W): 8 kg

	w	1	2	3	4	5	6	7	8
i	0	0	0	0	0	0	0	0	0
1	0	0	1	1	1	1	1	1	1
2	0	0	1	2	2	3	3	3	3
3	0	0	1	2	5	5	6	7	7
4	0	0	1	2	5	6	6	7	8

Example-2

$$G[i, w] = \begin{cases} 0 & \text{if } i = 0 \text{ or } w = 0 \\ G[i-1, w] & \text{if } w_i > w \\ \max\left(G[i-1, w],\ v_i + G[i-1, w - w_i]\right) & \text{otherwise} \end{cases}$$

i	1	2	3	4
v_i	1	2	5	6
w_i	2	3	4	5

Knapsack capacity (W): 8 kg

	w	1	2	3	4	5	6	7	8
i	0	0	0	0	0	0	0	0	0
1	0	0	1	1	1	1	1	1	1
2	0	0	1	2	2	3	3	3	3
3	0	0	1	2	5	5	6	7	7
4	0	0	1	2	5	6	6	7	8


Each of your actions will have an impact on your future.


Once you know who is walking with you on your path. you will never be afraid.

Thank you