Dynamic Programming 0-1 Knapsack Problem

Dr. P. Gayathri

Professor

SCOPE, VIT Vellore

Variations of the Knapsack problem

- **Fractional Knapsack - Fractions are allowed. Items are divisible.**
  - **This applies to items such as: Bread - taking half**
- **0-1 (or) Binary Knapsack - No fractions. Items are indivisible.**
  - **0/1 (1 brown pant, 1 green shirt…)**

0/1 Knapsack problem

- Given a knapsack with weight $W > 0$.
- A set S of n items with weights $w_i > 0$ and benefits $b_i > 0$ for $i = 1,\ldots,n$.
- $S = \{ (item_1, w_1, b_1), (item_2, w_2, b_2) , \ldots , (item_n, w_n, b_n) \}$
- Find a subset of the items which does not exceed the weight W of the knapsack and maximizes the benefit.

0/1 Knapsack problem

Determine a subset *A* of { 1, 2, …, *n* } that satisfies the following:

$$\max \sum_{i \in A} b_i \text{ where } \sum_{i \in A} w_i \leq W$$

In 0/1 knapsack a specific item is either selected or not

0/1 knapsack solution using DP

Let $W = 10$ and

i	1	2	3	4
v_i	10	40	30	50
w_i	5	4	6	3

$V[i, w]$	0	1	2	3	4	5	6	7	8	9	10
$i = 0$	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	10	10	10	10	10	10
2	0	0	0	0	40	40	40	40	40	50	50
3	0	0	0	0	40	40	40	40	40	50	70
4	0	0	0	50	50	50	50	90	90	90	90

Final Result

- {x1, x2, x3, x4} = {0, 1, 0, 1}
- Optimal profit is V[4,10] = 90
- Items selected: {2,4}

Basic Formula

- $V[i, w] = \text{Max}\ \{V[i-1, w]\ ,\ v[i] + V[i-1, w-w[i]\}$

Pseudocode

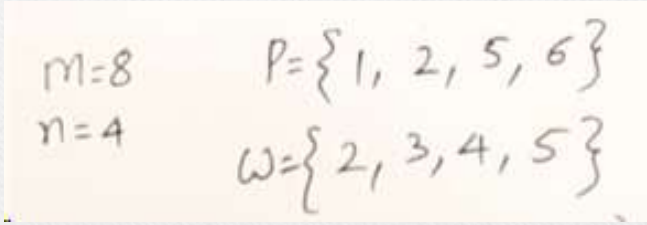
```
KnapSack(v, w, n, W)
{
    for (w = 0 to W) V[0, w] = 0;
    for (i = 1 to n)
        for (w = 0 to W)
            if ((w[i] ≤ w) and (v[i] + V[i − 1, w − w[i]] > V[i − 1, w]))
            {
                V[i, w] = v[i] + V[i − 1, w − w[i]];
                keep[i, w] = 1;
            }
            else
            {
                V[i, w] = V[i − 1, w];
                keep[i, w] = 0;
            }
    K = W;
    for (i = n downto 1)
        if (keep[i, K] == 1)
        {
            output i;
            K = K − w[i];
        }
    return V[n, W];
}
```

Time Complexity

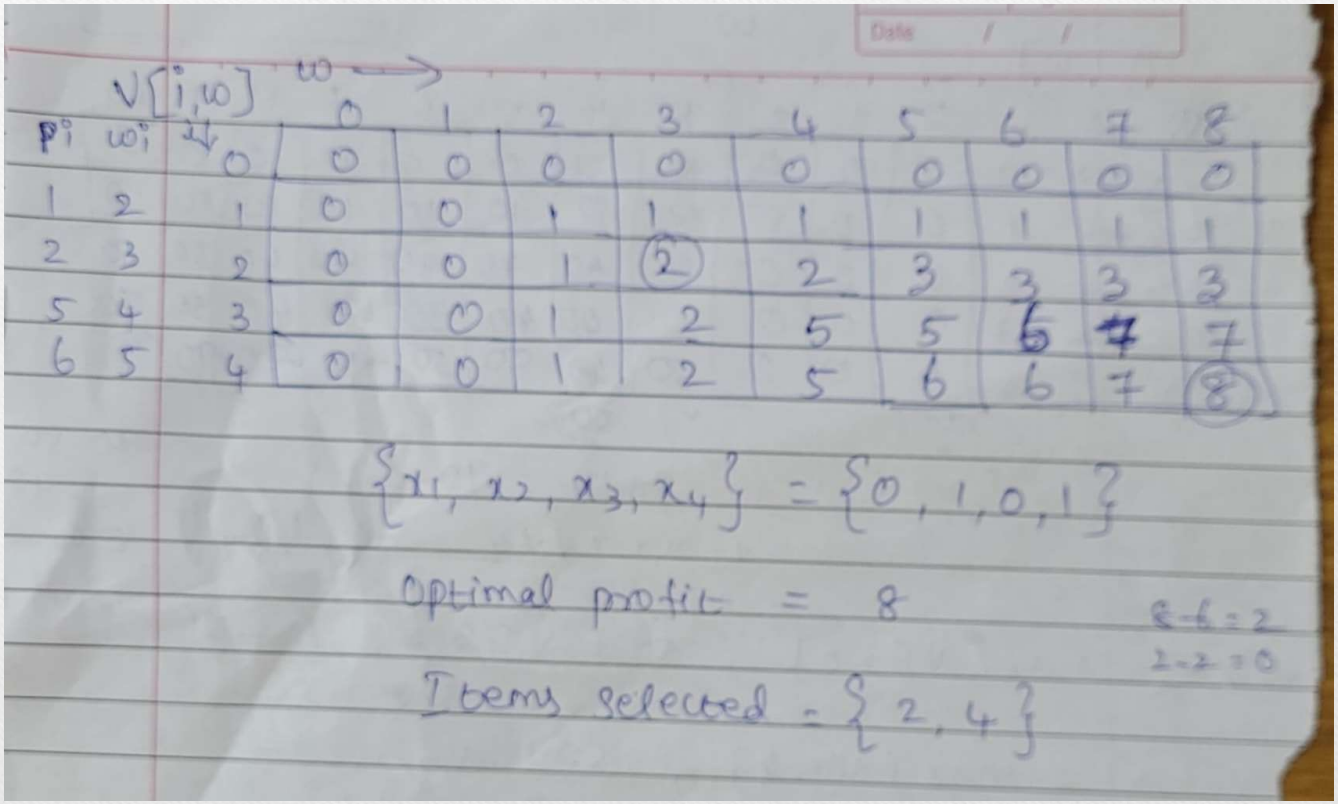
- Time complexity of 0-1 Knapsack problem is **O(nW)** where, n is the number of items and W is the capacity of knapsack.

Practice Problem

- Solve 0-1 knapsack problem using DP.



Solution



V[i, w]

pi	wi	↓ w →	0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	(2)	2	3	3	3	3
5	4	3	0	0	1	2	5	5	6	7	7
6	5	4	0	0	1	2	5	6	6	7	(8)

$\{x_1, x_2, x_3, x_4\} = \{0, 1, 0, 1\}$

Optimal profit = 8

8-6 = 2
2-2 = 0

Items selected = {2, 4}