Design and Analysis of Algorithms

Lecture-27

Dharmendra Kumar (Associate Professor)
Department of Computer Science and Engineering
United College of Engineering and Research,
Prayagraj

Knapsack problem

- A thief is robbing a store and can carry a maximal weight of W into his knapsack. There are n items available in the store and weight of i^{th} item is w_i and its profit is p_i. What items should the thief take?
- Given weights and values of n items, put these items in a knapsack of capacity W to get the maximum total value in the knapsack. In other words, given two integer arrays val[0..n-1] and wt[0..n-1] which represent values and weights associated with n items respectively. Also given an integer W which represents knapsack capacity, find out the maximum value subset of val[] such that sum of the weights of this subset is smaller than or equal to W.
- Based on the nature of the items, Knapsack problems are categorized as
  1. Fractional Knapsack
  2. 0-1 Knapsack

Knapsack problem

Fractional Knapsack

In this version of Knapsack problem, items can be broken into smaller pieces. So, the thief may take only a fraction x_i of i^{th} item.

0-1 Knapsack

In this version of Knapsack problem, we cannot break an item, either pick the complete item or don't pick it (0-1 property).

Knapsack problem

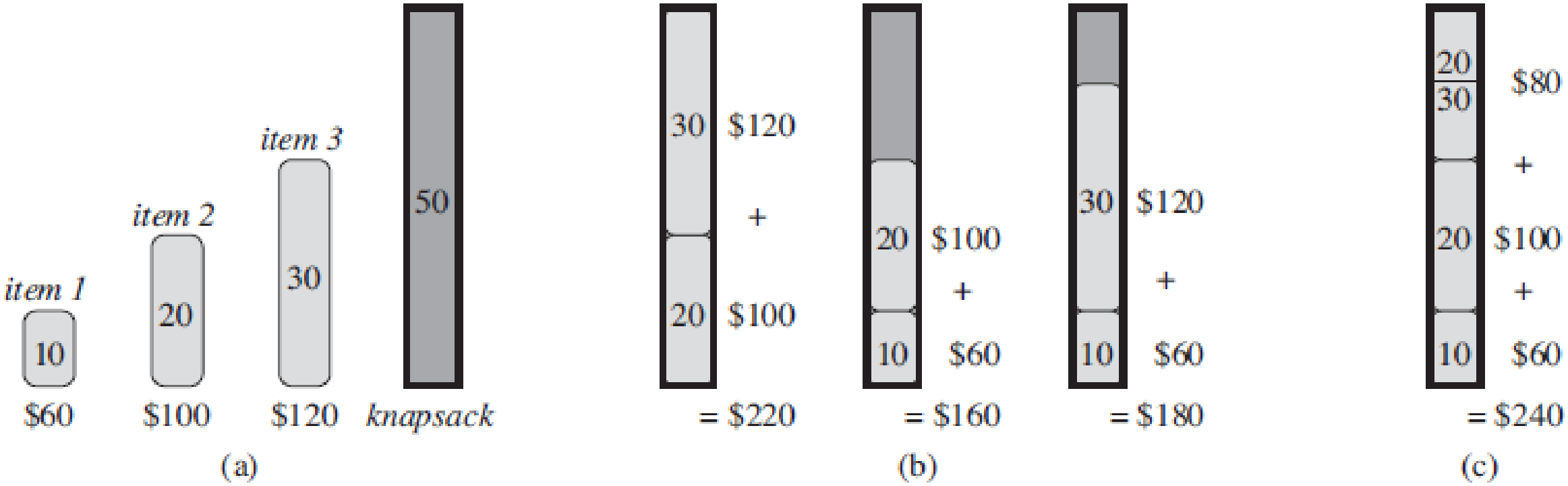
Greedy approach to solve the knapsack problem

- First, we compute value v_i/w_i for each item i.
- Arrange all the items in descending order on the basis of v_i/w_i .
- Put first item in the knapsack fully if weight of first item is less than or equal to W. Put first item in the knapsack partially if weight of first item is greater than W.
- Similarly, put the second item in the knapsack fully if there is a sufficient space in the knapsack. Otherwise, we put second item partially.
- This process continue till Knapsack is filled.

Knapsack problem

Note: **Greedy algorithm solves the fractional knapsack but 0-1 Knapsack can not be solved by greedy algorithm.**

To see that this greedy strategy does not work for the 0-1 knapsack



Knapsack problem

Example: Consider the following instance for knapsack problem. Find the solution using Greedy method:

N= 8, W=130

W = {21, 31, 43, 53, 41, 63, 65, 75}

V = {11, 21, 31, 33, 43, 53, 65, 65}

Solution: Compute

$$\frac{V}{W} = \left\{ \frac{11}{21}, \frac{21}{31}, \frac{31}{43}, \frac{33}{53}, \frac{43}{41}, \frac{53}{63}, \frac{65}{65}, \frac{65}{75} \right\}$$

$$= \{ 0.52, 0.68, 0.72, 0.62, 1.05, 0.84, 1, 0.87 \}$$

First arrange all items in descending order of their value per weight i.e. (v_i/w_i).

W' = {41,65,75,63,43,31,53,21}

V' = {43,65,65,53,31,21,33,11}

Descending order of items = item5, item7, item8, item6, item3, item2, item4, item1

Therefore, optimal solution = (0, 0, 0, 0, 1, 0, 1, 24/75)

Optimal value = 43 + 65 + (24/75)*65 = 108 + 20.8 = **128.8**

Knapsack problem

Example: Consider the following instance for knapsack problem. Find the solution using Greedy method:

w = (5, 10, 20, 30, 40), v = (30, 20, 100, 90,160)

The capacity of knapsack W = 60

Example: Given the six items in the table below and a Knapsack with Weight 100, what is the solution to the Knapsack problem in all concepts. i.e. explain greedy all approaches and find the optimal solution.

ITEM ID	WEIGHT	VALUE	VALUE/WEIGHT
A	100	40	.4
B	50	35	.7
C	40	20	.5
D	20	4	.2
E	10	10	1
F	10	6	.6

Knapsack problem

Algorithm: Assume all the items are arranged in descending order of their value per weight i.e. (v_i/w_i).

```
Fractional_Knapsack(w, v, W)
        n = length[w]
        for i=1 to n
         do      x[i] = 0
        i=1
        weight = 0
        while( I <= n and weight < W)
        do      if(weight+w[i] < = W)
                then    x[i] = 1
                        weight = weight + w[i]
                else
                        x[i] =  (W – weight)/w_i
                        weight = W
        return x
```

- Time complexity of this algorithm will be O(n) if all the items are arranged in descending order of their value per weight .

- Time complexity of this algorithm will be O(nlogn) if all the items are not arranged in descending order of their value per weight .