Design and Analysis of Algorithm (DAA)

Greedy Algorithms

(Fractional Knapsack)

Dr. Dayal Kumar Behera

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

What is Knapsack Problem?

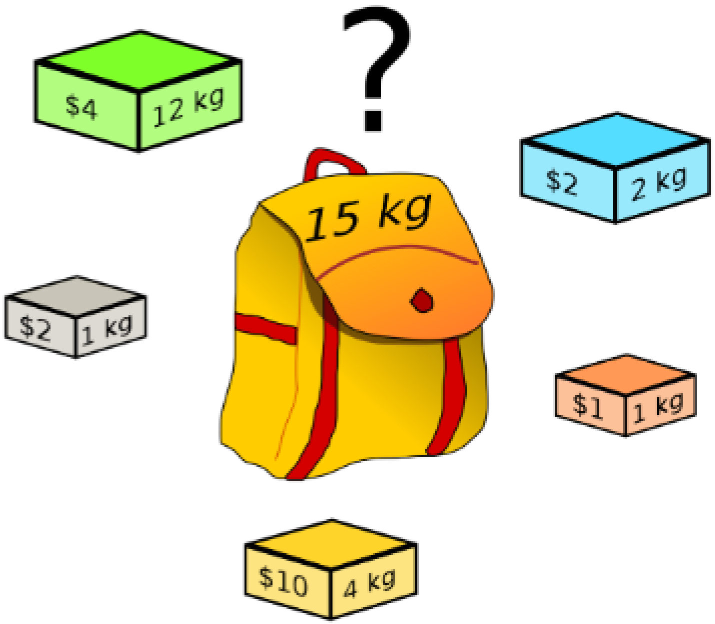
- **The classic Knapsack Problem:**

"A thief breaks into a store and wants to fill his knapsack of capacity W with goods of as much value as possible"

- **Decision version of the Problem:**

"Does there exist a collection of items that fits into his knapsack and whose total value is greater than or equal to a given W?"

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 most valuable subset of items with total weight less than or equal to knapsack capacity.*

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

Knapsack Problem Variants-

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

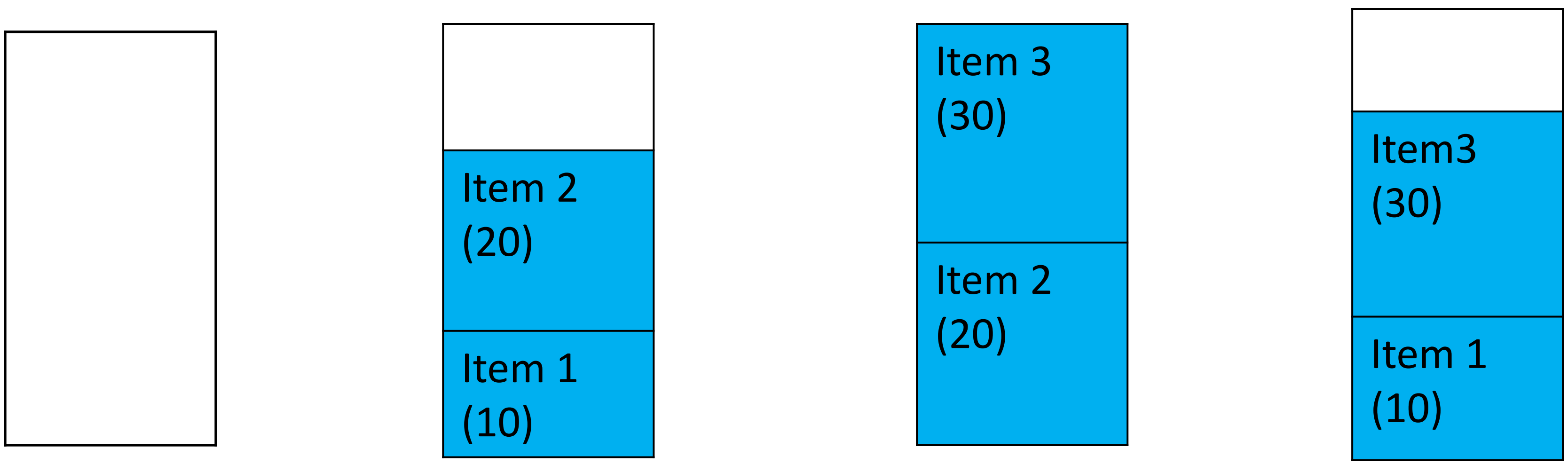
2. Fractional Knapsack Problem:

- 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

Example 1 (Fractional Knapsack)

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

Knapsack capacity (W): 50 kg

i	1	2	3
item	1	2	3
v_i	60	100	120
w_i	10	20	30
$\frac{v_i}{w_i}$	6	5	4

Item 3 (20)

Item 2 (20)

Item 1 (10)

Total Weight: 0
Profit: 0

Total Weight: 50
Profit: 240

- First compute value per unit or Profit Ratio (v/w)
- Sort items by profit ratio in descending order

Example 2 (Fractional Knapsack)

Item List with Weight and Profit

i	1	2	3	4	5
v_i	24	30	22	25	11
w_i	14	15	10	7	6

Total Weight Limit: 30

Greedy Component of solving Knapsack Problem:

1. **Selection Procedure:** Sorting Based on the Profit Ratio (v/w)
2. **Feasibility Check :** Whether the total weight is less than the capacity of Knapsack
3. **Solution Check:** Whether all the instances are checked or not

Example 2 (Fractional Knapsack)

i	1	2	3	4	5
v_i	24	30	22	25	11
w_i	14	15	10	7	6
v_i/w_i	1.714	2	2.2	3.571	1.83

Sorted

Based on Profit per unit weight

item	4	3	2	5	1
v_i	25	22	30	11	24
w_i	7	10	15	6	14
v_i/w_i	3.571	2.2	2	1.83	1.714

Solution Set

Index i	1	2	3	4	5
item	4	3	2	5	1
v_i	25	22	30	11	24
w_i	7	10	15	6	14
v_i/w_i	3.571	2.2	2	1.83	1.714
Selected Weight	7	10	13	x	x
Profit	25	22	26		

Total Weight Limit: 30

Total Profit: 73

Another Example

EXAMPLE :

(w1,p1)	(w2,p2)	(w3,p3)	(w4,p4)	(w5,p5)
(120,5)	(150,5)	(200,4)	(150,8)	(140,3)

Total capacity c = 600

The profit / weight ratios are :

<u>p1/w1 = 5/120 = .0417;</u> <u>p2/w2 = 5/150 = .0333;</u> <u>p3/w3 = 4/200 = .0200;</u>

<u>p4/w4 = 8/150 = .0533;</u> <u>p5/w5 = 3/140 = .0214;</u>

Thus the order of consideration is 4, 1, 2, 5, 3

Another Example

We take all of items 4, 1, 2 and 5 for a total weight of knapsack is

$$150 + 120 + 150 + 140 = 560$$

We now only have room for 40 units of weight, so we take 40/200 = 1/5 of object 3

The profit is thus $5 + 5 + 0.2 \cdot 4 + 8 + 3 = \boxed{21.8}$

Fractional Knapsack

Assumption: items are sorted by $\frac{v_i}{w_i}$ in descending order

```
Algorithm GREEDY_KNAPSACK(W, n)

for i ← 1 to n do
        x[i] = 0.0
//end for

load = 0
profit=0
```

```
i = 1
while ((load <W) and (i ≤ n)) do
        if( w_i ≤ ( W – load) ) then
                load = load + w_i
                x[i] = 1
        else
                r = W – load
                load = load + r
                x[i] = r/w_i
        end if
        profit= profit+ v[i]. x[i]
        i = i + 1
//end while
return x and profit
```

x : item-wise selected weight

Time Complexity

Assumption: items are sorted by $\frac{v_i}{w_i}$ in descending order

1. Calculate v_i / w_i for i = 1, 2, …, n **O(n)**
2. Sort items by nonincreasing v_i / w_i **O(nlogn)**

```
Algorithm GREEDY_KNAPSACK(W, n)

for i ← 1 to n do
        x[i] = 0.0
//end for

load = 0
profit=0
```

```
i = 1
while ((load <W) and (i ≤ n)) do
        if( w_i ≤ ( W – load) ) then
                load = load + w_i
                x[i] = 1
        else
                r = W – load
                load = load + r
                x[i] = r/w_i
        end if
        profit= profit+ v[i]. x[i]
        i = i + 1
//end while
return x and profit
```

O(n)

Overall:

O(nlogn)


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