X SECTION

12 March 2025 13:13

Greedy (लालची) --> Greed (लालच)

Greedy algorithms are a class of algorithms that make **locally optimal** choices at each step with the hope of finding a **global optimum** solution. Examples of popular Greedy Algorithms are Fractional Knapsack, Dijkstra's algorithm, Kruskal's algorithm, Huffman coding and Prim's Algorithm.

Greedy: Think about present (**वर्तमान**)and not future (**भविष्य**).

Dynamic Programming: Think about future and not present. Think if in future it will give optimum solution (best possible solution) or not.

Eg: Cloth shop. During Festival time, we get offers on clothes in different shops. So what is our tendency, we move to that shop where:

1. Price is low ignoring the quality of the cloth. This is greedy.

DP= Low price along with quality of cloth matters.

Fractional Knapsack Problem

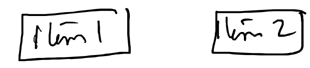
Given two arrays named **value** and **weight** of size n each, where **value[i]** represents the value and **weight[i]** represents the weight associated with the i^{th} item. Also, we have been provided a knapsack with a maximum capacity of W.

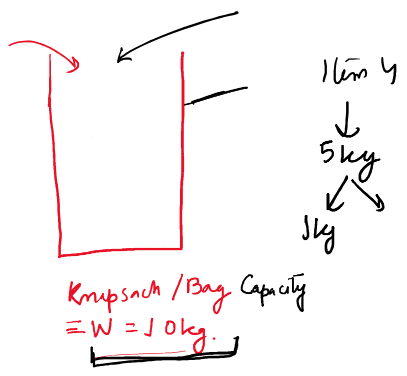
The task is to pick some items (possibly all of them) such that the sum of values of all the items is maximized and the sum of weights of all the items is at most W.

As the name `Fractional` suggests, here we are allowed to take a fraction of an item.

For an item of weight 20 Kg. we can either take 1 Kg, 2 Kg, ..., or 19 Kg but we are not allowed to take 1.2 Kg or 5.7

Item	1	2	3	4	5
Value/Profit	10	15	10	20	8
Weight	3	3	2	5	1





So, we can opt for a greedy algorithm. We can have several potentially correct strategies, some of the obvious ones are :

1. Picking the items with the largest values first.
2. Picking the items with the lowest weights first.
3. Picking the items based on some sort of ratio among their values and weights.

Need to consider both profit and weight at the same time. Value DIVIDE by weight gives the profit for 1 kg of item.

1.
Object |||| profit |||| weight |||| remaining_weight

1. Greedy about profit: Means place those items first in the bag/knapsack which has more profit. This will in turn maximise our profit. जिस object का सबसे ज्यादा मूल्य है, उसे बैग के अंदर डालूंगा।

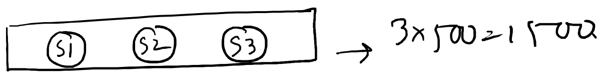
Item	1	2	3	4	5
Value/Profit	10	15	10	20	8
Weight	3	3	2	5	1

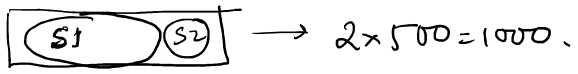
Knapsack capacity = 10kg.

Item	Profit	Weight	Remaining weight
4	20	5	10-5=5 kg.
2	15	3	5-3=2kg.
3	10	2	2-2=0kg.
	45		

Greedy about weight: Means place those items first in the bag/knapsack which has least weight. This will help us to put more items in our bag which will in turn maximise our profit. अगर मैं कम वजन के वस्तुओं को बैग के अंदर डालूंगा, तो मैं ज्यादा वस्तुएं बैग के अंदर डाल सकता हूँ।

Tuition Center - 1 student = Rs 500 fee.





Item	Profit	Weight	Remaining weight
5	8	1	10-1=9
3	10	2	9-2=7.
2	15	3	7-3=4.
1	10	3	4-3=1
5	20/5=4	1	1-1=0.
	47.		

Item	1	2	3	4	5
Value/Profit	10	15	10	20	8
Weight	3	3	2	5	1

③ Greedy about profit && weight (both).

Item	1	2	3	4	5
Value/Profit	10	15	10	20	8
Weight	3	3	2	5	1
1 kg price	3.33	5	5	4	8

✓ ① Find price of 1 kg for each items.

② Sort in descending order of $\frac{\text{Profit}}{\text{Weight}}$.

Item	5	2	3	4	1
Val	8	15	10	20	10
Wt	1	3	2	$\boxed{5}$	3

5 kg $\rightarrow$ Rs 20

1 kg $\rightarrow$ Rs 4

4 kg $\rightarrow$ Rs 16.

Item	Profit	Wt	Remaining Wt.
5	8	1	10-1=9.
2	15	3	9-3=6
3	10	2	6-2=4.
4	16	4	4-4=0
	49		

Arrange all the items in descending order based on P_i/W_i

Items	5	2	3	4	1
Weights (in kg)	1	3	2	5.	3
Profits	8	15	10	20	10
P_i/W_i (Price of 1 kg).	8	5	5	4	3.3

Item	Profit	Weight	Remaining weight

Greedy	DP (Dynamic Programming).

Greedy	DP (Dynamic Programming):
① At each step you want optimum solution (best), with the hope of finding the optimum solution in the end.	① Aim → Final solution will be optimum.

<u>Shopping</u> =

$$\begin{tikzcd}
 & \text{Shopping} \arrow[dl] \arrow[dr] & \\
\text{Shop 1} & & \text{Shop 2} \\
\text{50\% Discount} \arrow[d, "\text{Greedy}"] & & \text{20\% Discount} \arrow[d, "\text{DP}"] \\
\text{Present} & & \text{Future.}
\end{tikzcd}$$