Greedy Algorithm Introduction

"Greedy Method finds out of many options, but you have to choose the best option."

In this method, we have to find out the best method/option out of many present ways.

In this approach/method we focus on the first stage and decide the output, don't think about the future.

This method may or may not give the best output.

Greedy Algorithm solves problems by making the best choice that seems best at the particular moment. Many optimization problems can be determined using a greedy algorithm. Some issues have no efficient solution, but a greedy algorithm may provide a solution that is close to optimal. A greedy algorithm works if a problem exhibits the following two properties:

1. **Greedy Choice Property:** A globally optimal solution can be reached at by creating a locally optimal solution. In other words, an optimal solution can be obtained by creating "greedy" choices.
2. **Optimal substructure:** Optimal solutions contain optimal subsolutions. In other words, answers to subproblems of an optimal solution are optimal.

Example:

1. machine scheduling
2. Fractional Knapsack Problem
3. Minimum Spanning Tree
4. Huffman Code
5. Job Sequencing
6. Activity Selection Problem

Steps for achieving a Greedy Algorithm are:

1. **Feasible:** Here we check whether it satisfies all possible constraints or not, to obtain at least one solution to our problems.
2. **Local Optimal Choice:** In this, the choice should be the optimum which is selected from the currently available
3. **Unalterable:** Once the decision is made, at any subsequence step that option is not altered.

Fractional Knapsack

Fractions of items can be taken rather than having to make binary (0-1) choices for each item.

Fractional Knapsack Problem can be solvable by greedy strategy whereas 0 - 1 problem is not.

Steps to solve the Fractional Problem:

1. Compute the value per pound v_i/w_i for each item.
2. Obeying a Greedy Strategy, we take as possible of the item with the highest value per pound.
3. If the supply of that element is exhausted and we can still carry more, we take as much as possible of the element with the next value per pound.
4. Sorting, the items by value per pound, the greedy algorithm run in O (n log n) time.

```
Fractional Knapsack (Array v, Array w, int W)
1. for i= 1 to size (v)
2. do p [i] = v [i] / w [i]
3. Sort-Descending (p)
4. i ← 1
5. while (W>0)
6. do amount = min (W, w [i])
7. solution [i] = amount
8. W= W-amount
9. i ← i+1
10. return solution
```

Example: Consider 5 items along their respective weights and values: -

$I = (I_1, I_2, I_3, I_4, I_5)$

w = (5, 10, 20, 30, 40)

v = (30, 20, 100, 90,160)

The capacity of knapsack W = 60

Now fill the knapsack according to the decreasing value of p_i.

First, we choose the item I_i whose weight is 5.

Then choose item I_3 whose weight is 20. Now,the total weight of knapsack is 20 + 5 = 25

Now the next item is I_5, and its weight is 40, but we want only 35, so we chose the fractional part of it,

$$\text{i.e., } 5 \times \frac{5}{5} + 20 \times \frac{20}{20} + 40 \times \frac{35}{40}$$

$$\text{Weight} = 5 + 20 + 35 = 60$$

Maximum Value:-

$$30 \times \frac{5}{5} + 100 \times \frac{20}{20} + 160 \times \frac{35}{40}$$

$$= 30 + 100 + 140 = 270 \text{ (Minimum Cost)}$$

Solution:

ITEM	w_i	v_i
I_1	5	30
I_2	10	20
I_3	20	100
I_4	30	90
I_5	40	160

Taking value per weight ratio i.e. $p_i = \frac{v_i}{w_i}$

ITEM	w_i	v_i	$P_i = \frac{v_i}{w_i}$
I_1	5	30	6.0
I_2	10	20	2.0

I_3	20	100	5.0
I_4	30	90	3.0
I_5	40	160	4.0

Now, arrange the value of p_i in decreasing order.

ITEM	w_i	v_i	$p_i = \frac{v_i}{w_i}$
I_1	5	30	6.0
I_3	20	100	5.0
I_5	40	160	4.0
I_4	30	90	3.0
I_2	10	20	2.0

Example

Let us consider that the capacity of the knapsack ***W* = 60** and the list of provided items are shown in the following table −

Item	A	B	C	D
Profit	280	100	120	120
Weight	40	10	20	24
Ratio (pi/wi)	7	10	6	5

As the provided items are not sorted based on pi/wi. After sorting, the items are as shown in the following table.

Item	B	A	C	D
Profit	100	280	120	120
Weight	10	40	20	24
Ratio (pi/wi)	10	7	6	5

Solution

After sorting all the items according to pi/wi. First all of ***B*** is chosen as weight of ***B*** is less than the capacity of the knapsack. Next, item ***A*** is chosen, as the available capacity of the knapsack is greater than the weight of ***A***. Now, ***C*** is chosen as the next item. However, the whole item cannot be chosen as the remaining capacity of the knapsack is less than the weight of ***C***.

Hence, fraction of ***C*** (i.e. (60 − 50)/20) is chosen.

Now, the capacity of the Knapsack is equal to the selected items. Hence, no more item can be selected.

The total weight of the selected items is **10 + 40 + 20 * (10/20) = 60**

And the total profit is **100 + 280 + 120 * (10/20) = 380 + 60 = 440**

This is the optimal solution. We cannot gain more profit selecting any different combination of items.