Fractional Knapsack Problem using Greedy Approach

Problem Scenario :

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?

In this context, the items should be selected in such a way that the thief will carry those items for which he will gain maximum profit. Hence, the objective of the thief is to maximize the profit.

Based on the nature of the items, Knapsack problems are categorized as

- Fractional Knapsack
- 0 1 Knapsack

Fractional Knapsack

In this case, items can be broken into smaller pieces, hence the thief can select fractions of items.

According to the problem statement,

- There are **n** items in the store
- Weight of i^{th} item $w_i>0$
- Profit for i^{th} item $p_i>0$ and
- Capacity of the Knapsack is **W**

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.

$$0\leqslant x_i\leqslant 1$$

The i^{th} item contributes the weight $x_i.w_i$ to the total weight in the knapsack and profit $x_i.p_i$ to the total profit.

Hence, the objective of this algorithm is to

$$\text{maximize } \sum_{i=1}^{n}(x_i.p_i)$$

subject to constraint,

$$\sum_{i=1}^{n}(x_i.w_i)\leqslant W$$

It is clear that an optimal solution must fill the knapsack exactly, otherwise we could add a fraction of one of the remaining items and increase the overall profit.

Thus, an optimal solution can be obtained by

$$\sum_{i=1}^{n}(xi.wi)=\mathrm{W}$$

In this context, first we need to sort those items according to the value of p_i/w_i, so that $p_i+1/w_i+1 \le p_i/w_i$. Here, ***x*** is an array to store the fraction of items.

```
Algorithm: Greedy-Fractional-Knapsack (w[1..n], p[1..n], W)
for i = 1 to n
   do x[i] = 0
weight = 0
profit = 0
for i = 1 to n
   if weight + w[i] ≤ W then
      x[i] = 1
      weight = weight + w[i]
      profit = profit + p[i]
   else
      x[i] = (W - weight) / w[i]
      weight = W
      profit = profit + p[i] * x[i]
      break
```

→ Array having profits of items

Array having weights of items

Knapsack Capacity

→ Array having parts of taken items

$(60-50)/20 = \frac{10}{20} = \frac{1}{2}$

$380 + 120 \times \frac{1}{2} = 380 + 60$
$= 440$

Analysis

If the provided items are already sorted into a decreasing order of p_i/w_i, then the loop takes a time in ***O(n)***; Therefore, the total time including the sort is in ***O(n logn)***.

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 (p_i/w_i)	7	10	6	5

As the provided items are not sorted based on p_i/w_i. 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 (p_i/w_i)	10	7	6	5

W = 60

B → 10 ✓ → 100

Solution

A → 40 ✓ → 280

After sorting all the items according to p_i/w_i. 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***.

C → 10 ✓ → 60

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.

440 → Total profit

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.