

# CTA PRACTICAL 6 20MCED08

### **Practical 6**

Implement fractional knapsack problem using greedy approach.

### Introduction

Given a set of items, each with a weight and a value, determine a subset of items to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

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

- Fractional Knapsack
- Knapsack

## Fractional Knapsack

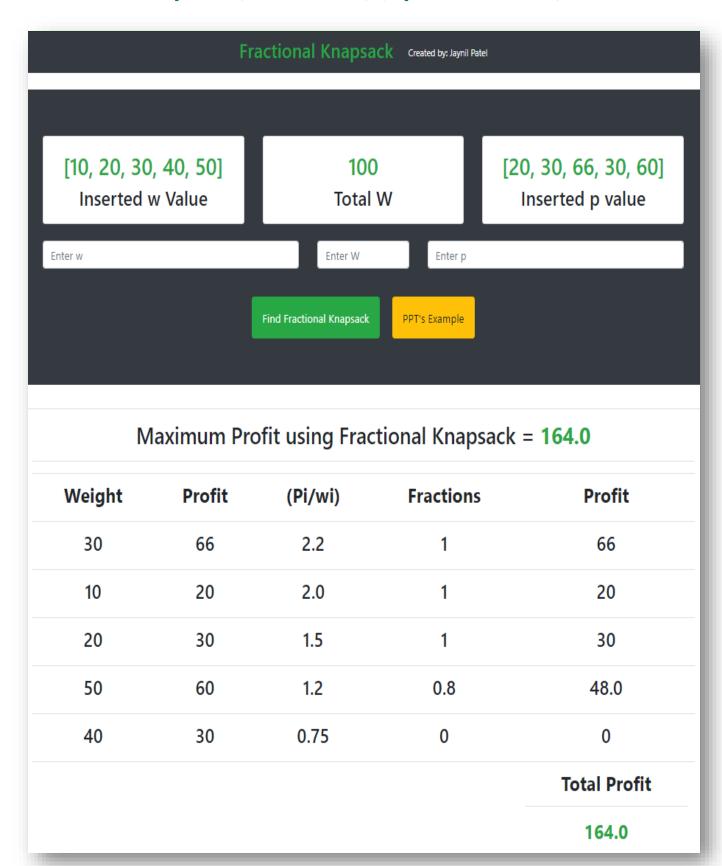
In this case, items can be broken into smaller pieces, hence the thief can select fractions of items.

According to the problem statement,

- There are **n** items in the store
- Weight of ith item wi>0
- Profit for  $i^{th}$  item pi>0 and
- Capacity of the Knapsack is W
- N=5, W=100

w	10	20	30	40	50
V	20	30	66	40	60
v/w	2.0	1.5	2.2	1.0	1.2

# Fractional Knapsack(Simulation) (Python + Flask)



# Fractional Knapsack(Python)

```
def FK(w,p,W):
    values = [[0,0,0] for i in w]
    profit =[0 for i in w]
    x = [0 \text{ for i in } w]
    for i in range(0,len(w)):
        values[i][0] = w[i]
        values[i][1] = p[i]
        values[i][2] = p[i] / w[i]
    values.sort(key=lambda x :x[2],reverse=True)
    weight = 0
    for i in range(0,len(w)):
        if values[i][0] + weight <= W:</pre>
            x[i] = 1
            profit[i] = values[i][1]
            weight = weight + values[i][0]
            temp = (W - weight) / values[i][0]
            x[i] = temp
            weight = W
            profit[i] = temp * values[i][1]
    print("\nMaximum Profit = ",sum(profit))
    print("\nWeight\t Value \t Ratio \t Fraction\n")
    for values,j in zip(values,x):
        print(values[0],"\t",values[1],"\t",values[2],"\t",j)
W = [10, 20, 30, 40, 50]
p = [20, 30, 66, 40, 60]
W = 100
FK(w,p,W)
# By Jaynil Patel (20MCED08)
```

### **Output:**

Maximum Profit = 164.0

Weight	Value	Ratio	Fraction
30	66	2.2	1
10	20	2.0	1
20	30	1.5	1
50	60	1.2	0.8
40	40	1.0	0

# **Observation:**

If the provided items are already sorted into a decreasing order of V/W, then the loop ta kes a time in O(n).

Therefore, the total time including the sort is in *O(n logn)*.