# Department of Computer Science and Engineering (Data Science)

## Experiment 4

## (Greedy Algorithm)

**Aim:** Implementation of fractional Knapsack using greedy algorithm.

## Theory:

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.

The knapsack problem is in combinatorial optimization problem. It appears as a subproblem in many, more complex mathematical models of real-world problems. One general approach to difficult problems is to identify the most restrictive constraint, ignore the others, solve a knapsack problem, and somehow adjust the solution to satisfy the ignored constraints.

Applications:

In many cases of resource allocation along with some constraint, the problem can be derived in a similar way of Knapsack problem. Following is a set of example.

- Finding the least wasteful way to cut raw materials
- portfolio optimization
- Cutting stock problems

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
-

## Pseudocode:

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

```
  else
    x[i] = (W - weight) / w[i]
    weight = W
    break
return x
```

## Complexity:

Time Complexity: O(n logn).

## Example:

Problem: Consider the following instances of the fractional knapsack problem: n = 3, M = 20, V = (24, 25, 15) and W = (18, 15, 20) find the feasible solutions.

Solution:

Arrange items by decreasing order of profit density. Assume that items are labeled as X = (I1, I2, I3), have profit V = {24, 25, 15} and weight W = {18, 15, 20}.

| **Item ($x_i$)** | **Value ($v_i$)** | **Weight ($w_i$)** | $p_i = v_i / w_i$ |
|---|---|---|---|
| $I_2$ | 25 | 15 | 1.67 |
| $I_1$ | 24 | 18 | 1.33 |
| $I_3$ | 15 | 20 | 0.75 |

Initialize, Weight of selected items, SW = 0,

Profit of selected items, SP = 0,

Set of selected items, S = { },

Here, Knapsack capacity M = 20.

Iteration 1 : SW= (SW + $w_2$) = 0 + 15 = 15

SW ≤ M, so select $I_2$

S = { $I_2$ }, SW = 15, SP = 0 + 25 = 25

Iteration 2 : SW + $w_1$ > M, so break down item $I_1$.

The remaining capacity of the knapsack is 5 unit, so select only 5 units of item $I_1$.

frac = (M – SW) / W[i] = (20 – 15) / 18 = 5 / 18

S = { $I_2$, $I_1$ * 5/18 }

SP = SP + $v_1$ * frac = 25 + (24 * (5/18)) = 25 + 6.67 = 31.67

SW = SW + $w_1$ * frac = 15 + (18 * (5/18)) = 15 + 5 = 20

The knapsack is full. Fractional Greedy algorithm selects items {$I_2$, $I_1$ * 5/18 }, and it gives a profit of **31.67 units**.

## Lab Assignment to Complete:

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 |

**Solution:**

**Code:**

**Output:**