Fractional Knapsack Problem

The Fractional Knapsack Problem is an optimization problem where we must maximize profit by selecting items with given weights and values, such that the total weight does not exceed the capacity of the knapsack.

Unlike the 0/1 Knapsack, we can take fractions of items.

Greedy Strategy Used:

1. Compute value/weight ratio for each item.
2. Sort items in decreasing order of this ratio.
3. Add items to the knapsack in that order:
    - If the item fits completely, take it all.
    - Otherwise, take the fraction that fits.
4. Stop when the knapsack is full.

Formula:

$$\text{Profit} = \sum(\text{value per weight}) \times \text{weight taken}$$

Complexity:

- Time Complexity: $O(n\log\ n)$(due to sorting)
- Space Complexity: $O(1)$

Code

```
# Fractional Knapsack Problem using Greedy Method
def fractional_knapsack(value, weight, capacity):
    n = len(value)
    ratio = [(value[i] / weight[i], value[i], weight[i]) for i in range(n)]
    ratio.sort(reverse=True)  # sort by value/weight ratio in descending order
    total_value = 0.0
```

```
    for r, v, w in ratio:
        if capacity >= w:
            capacity -= w
            total_value += v
        else:
            total_value += r * capacity
            break
    return total_value
# ----- MAIN PROGRAM -----
n = int(input("Enter number of items: "))
value = []
weight = []
for i in range(n):
    v = float(input(f"Enter value of item {i+1}: "))
    w = float(input(f"Enter weight of item {i+1}: "))
    value.append(v)
    weight.append(w)
capacity = float(input("Enter capacity of knapsack: "))
max_value = fractional_knapsack(value, weight, capacity)
print("\nMaximum value that can be obtained =", max_value)
```