

Session 2

Fractional Knapsack Problem

The fractional knapsack problem is a variation of the classic knapsack problem where you are allowed to take fractions of items rather than having to take whole items. The goal is to maximize the total value of the items you can carry in a knapsack with a given weight capacity. It's a greedy algorithm problem that can be solved efficiently by following a specific approach.

Solution Approach

✗ Do not copy. Read for understanding and the viva

1. **Calculate Value-to-Weight Ratio:** For each item, calculate the value-to-weight ratio, which is the value of the item divided by its weight. This ratio helps to determine which items are more valuable per unit of weight.
2. **Sort Items:** Sort the items in descending order based on their value-to-weight ratio. This ensures that you consider the most valuable items first.
3. **Initialize Variables:** Initialize a variable to keep track of the total value and the remaining weight capacity of the knapsack.
4. **Iterate Through Sorted Items:** Iterate through the sorted list of items. For each item, check if it can fit in the remaining weight capacity of the knapsack.
 - If the item can fit completely, add its full value to the total value and decrease the remaining weight capacity accordingly.
 - If the item cannot fit completely, take the fraction of the item that can fit in the remaining weight capacity, add the corresponding fraction of the value to the total value, and set the remaining weight capacity to zero (since the knapsack is now full).
5. **Return Total Value:** After iterating through all the items, return the total value of the items in the knapsack.

Example

✗ Do not copy. Read for understanding and the viva

Suppose you have the following items with their weights and values:

Item	Weight	Value
1	10	60
2	20	100
3	30	120

And the knapsack has a weight capacity of 50. The value-to-weight ratios for the items are:

- Item 1: $60/10 = 6$
 - Item 2: $100/20 = 5$
 - Item 3: $120/30 = 4$
- Sorting the items by their value-to-weight ratio gives us:

1. Item 1 (6)
 2. Item 2 (5)
 3. Item 3 (4)
- Now, we can start filling the knapsack:

- Item 1 can fit completely (weight 10), so we add its full value (60) to the total value and reduce the remaining capacity to 40.
- Item 2 can also fit completely (weight 20), so we add its full value (100) to the total value and reduce the remaining capacity to 20.
- Item 3 cannot fit completely (weight 30), but we can take a fraction of it. We can take $20/30$ of item 3, which gives us a value of $(20/30) * 120 = 80$. We add this to the total value and reduce the remaining capacity to 0. The total value in the knapsack is 60 (from item 1) + 100 (from item 2) + 80 (from item 3) = 240. Therefore, the maximum value that can be obtained with the given weight capacity is 240.

Time Complexity

The time complexity of the fractional knapsack problem is $O(n \log n)$ due to the sorting step, where n is the number of items. The rest of the operations (calculating ratios and iterating through the items) take $O(n)$ time.

Objectives

✕ Do not copy. Read for understanding and the viva

The main objectives of this session are to:

- Implement Fractional Knapsack Problem
- Test the implementation on different problem instances
- Implement greedy technique in general

Problem Statement

■ Write in lab record

Implement Fractional Knapsack Algorithm and find out optimal result for the following problem instances:

Question 1

$(P_1, P_2, P_3, P_4, P_5, P_6, P_7) = (15, 5, 20, 8, 7, 20, 6)$

$(W_1, W_2, W_3, W_4, W_5, W_6, W_7) = (3, 4, 6, 8, 2, 2, 3)$

Maximum Knapsack Capacity = 18

Question 2

$(P_1, P_2, P_3, P_4, P_5) = (20, 30, 40, 32, 55)$

$(W_1, W_2, W_3, W_4, W_5) = (5, 8, 10, 12, 15)$

Maximum Knapsack Capacity = 20

Question 3

$(P_1, P_2, P_3, P_4, P_5, P_6, P_7) = (12, 10, 8, 11, 14, 7, 9)$

$(W_1, W_2, W_3, W_4, W_5, W_6, W_7) = (4, 6, 5, 7, 3, 1, 6)$

Maximum Knapsack Capacity = 15

Implementation

■ Write in lab record

■

Lab record: write one language only. Pick yours once and every page opens on it; the other tabs are the same solution for comparison.

Python

Knapsack.py

```
1 # Fractional Knapsack in Python
2 def get_max_value(profits, weights, capacity):
3     # Calculate profit/weight ratio and keep track of original index
4     items = []
5     for i in range(len(profits)):
6         items.append(
7             {
8                 "id": i + 1,
9                 "profit": profits[i],
10                "weight": weights[i],
11                "ratio": profits[i] / weights[i],
12            }
13        )
14
15    # Sort items by ratio in descending order (Greedy Strategy)
16    items.sort(key=lambda x: x["ratio"], reverse=True)
17
18    total_value = 0.0
19    for item in items:
20        if capacity > 0 and capacity ≥ item["weight"]:
21            # Take the full item
22            capacity -= item["weight"]
23            total_value += item["profit"]
24        elif capacity > 0:
25            # Take a fraction of the item to fill remaining capacity
26            total_value += item["profit"] * (capacity / item["weight"])
27            capacity = 0
28            break
29
30    return total_value
31
32
33 # Solving Question 1
34 p1, w1, c1 = [15, 5, 20, 8, 7, 20, 6], [3, 4, 6, 8, 2, 2, 3], 18
35 print(f"Result Q1: {get_max_value(p1, w1, c1):.2f}")
```

Conclusion

✖ Do not copy. Read for understanding and the viva

The fractional knapsack problem is a fundamental problem in combinatorial optimization that can be efficiently solved using a greedy algorithm. By calculating the value-to-weight ratio and sorting the items accordingly, we can maximize the total value of the items in the knapsack while respecting the weight capacity. This problem has practical applications in various fields such as resource allocation, budgeting, and logistics. Understanding the fractional knapsack problem and its solution approach is essential for solving more complex optimization problems in computer science and operations research.

[✎ Edit this page](#)

[< Previous](#)
Session 1

Session 3 [Next >](#)