Vidyavardhini's College of Engineering & Technology

Department of Computer Science and Engineering (Data Science)

ACADEMIC YEAR: 2024-25

Course: Analysis of Algorithm Lab

Course code: CSL401

Year/Sem: SE/IV

Experiment No.: 05
Aim: To implement fractional knapsack problem using greedy technique
Name: SOHAM HEMENDRA RAUT
Roll Number: 24
Date of Performance: 13/02/2025
Date of Submission: 06/03/2025

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Performance	5	
Understanding	5	
Journal work and timely submission.	10	
Total	**20**	

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Performance	5	3	2
Understanding	5	3	2
Journal work and timely submission.	10	8	4

Checked by

Name of Faculty : Mrs. Komal Champanerkar

Signature :

Date :

- **Aim: To implement fractional knapsack problem using greedy technique**
- **Theory:**

Given the weights and values of N items, in the form of {value, weight} put these items in a knapsack of capacity W to get the maximum total value in the knapsack.

In Fractional Knapsack, we can break items for maximizing the total value of the knapsack

The basic idea of the greedy approach is to calculate the ratio value/weight for each item and sort the item on the basis of this ratio.

Then take the item with the highest ratio and add them until we can't add the next item as a whole and at the end add the next item as much as we can.

Which will always be the optimal solution to this problem.

- **Algorithm:**

Step 1: Input

Read the number of items n. For each item, input its value and weight.

Input the knapsack's capacity C.

Step 2: Calculate Value-to-Weight Ratio:

$$\text{Ratio} = \frac{\text{Value}}{\text{Weight}}$$

For each item, compute the ratio:

Step 3: Sort Items:

Sort items in descending order by their value-to-weight ratio.

Step 4: Greedy approach:

Initialize total value = 0, remaining capacity = CCC.

For each item:

If the item's weight fits within remaining capacity, add its full value.

If not, take the fractional value of the item to fill the knapsack and set remaining capacity to 0.

Step 5: Return the total value of the knapsack.

❖ **Program:**

```
class Item:
    def __init__(self, value, weight):
        self.value = value
        self.weight = weight
        self.ratio = value / weight  # Value to weight ratio

def fractional_knapsack(capacity, items):
    # Sort items by value-to-weight ratio in descending order
    items.sort(key=lambda x: x.ratio, reverse=True)

    total_value = 0.0
    remaining_capacity = capacity

    for item in items:
        if remaining_capacity == 0:
            break

        if item.weight <= remaining_capacity:
            # Take the whole item
            total_value += item.value
            remaining_capacity -= item.weight
        else:
            # Take a fraction of the item
            total_value += item.value * (remaining_capacity /
item.weight)
            remaining_capacity = 0

    return total_value

def get_user_input():
    n = int(input("Enter the number of items: "))
    items = []

    for i in range(n):
        value = float(input(f"Enter the value of item {i+1}: "))
```

```
        weight = float(input(f"Enter the weight of item {i+1}: "))
        items.append(Item(value, weight))

    capacity = float(input("Enter the capacity of the knapsack: "))
    return capacity, items

if __name__ == "__main__":
    capacity, items = get_user_input()  # Get input from user
    max_value = fractional_knapsack(capacity, items)  # Solve the
fractional knapsack problem
    print(f"Maximum value that can be obtained: {max_value:.2f}")
```

❖ **Output:**

```
PS C:\Users\SOHAM> & C:/Users/SOHAM/anaconda3/python.exe c:/Users/SOHAM/.conda/knapsack.py
Enter the number of items: 7
Enter the value of item 1: 5
Enter the weight of item 1: 1
Enter the value of item 2: 10
Enter the weight of item 2: 3
Enter the value of item 3: 15
Enter the weight of item 3: 5
Enter the value of item 4: 7
Enter the weight of item 4: 4
Enter the value of item 5: 8
Enter the weight of item 5: 1
Enter the value of item 6: 9
Enter the weight of item 6: 3
Enter the value of item 7: 4
Enter the weight of item 7: 2
Enter the capacity of the knapsack: 15
Maximum value that can be obtained: 51.00
PS C:\Users\SOHAM>
```

❖ **Conclusion:** The Fractional Knapsack Problem is solved using the Greedy Algorithm with O(N log N) complexity due to sorting, followed by O(N) for iteration, making the total complexity **O(N log N).**