ARTIFICIAL INTELLIGENCE (18CSC305J) LAB

EXPERIMENT 2

GREEDY APPROACH

Harsh Goel
RA1811003010185
CSE-C1

Aim:

To solve Knapsack Problem using Greedy Approach

Problem Description:

Given weights and values of n items, we need to 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 knapsack. This problem in which we can break an item is also called the fractional knapsack problem

Problem Formulation:

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

A simple code with our own comparison function can be written as follows, please see sort function more closely, the third argument to sort function is our comparison function which sorts the item according to value/weight ratio in non-decreasing order.

After sorting we need to loop over these items and add them in our knapsack satisfying above-mentioned criteria.

Algorithm:

STEP 1. The function fractional_knapsack is defined.

STEP 2. It takes three arguments: two lists value and weight; and a number capacity.

STEP 3. It returns (max_value, fractions) where max_value is the maximum value of items with total weight not more than capacity.

STEP 4. fractions is a list where fractions[i] is the fraction that should be taken of item i, where 0 <= i < total number of items.

STEP 5. The function works by choosing an item from the remaining items that has the maximum value to weight ratio.

STEP 6. If the knapsack can include the entire weight of the item, then the full amount of the item is added to the knapsack.

STEP 7. If not, then only a fraction of this item is added such that the knapsack becomes full.

STEP 8. The above three steps are repeated until the knapsack becomes full, i.e. the total weight reaches the maximum weight.

Source Code:

Language-**PYTHON**

```
def fractional_knapsack(value, weight, capacity):
    index = list(range(len(value)))  # index for looping
    ratio = [v/w for v, w in zip(value, weight)]
    index.sort(key=lambda i: ratio[i], reverse=True)
    max_value = 0
    fractions = [0]*len(value)  # array of 0s  == length of the value (n
o of values entered ie n)
    for i in index:
        if weight[i] <= capacity: # check if weight less than cap if yes
, 1 fraction is added
            fractions[i] = 1
            max_value += value[i] # addd to max value which can be carri
ed
            capacity -= weight[i] # reduce the capacity
        else:
            fractions[i] = capacity/weight[i]    # if we need to add in
fractions
            max_value += value[i]*capacity/weight[i]  # add the fraction
 to the max_value
            break

    return max_value, fractions
    #  max_value =>  maximum value of items with total weight not more t
han capacity
    #  fractions =>  fractions in which the items has to be carried


n = int(input('Number of items: '))
```

```
value = input('Enter the values of the ' + str(n) + ' item(s) in order:
').split()

weight = input('Enter the positive weights of the ' + str(n) + ' item(s)
 in order: ').split()

capacity = int(input('Enter maximum weight that can be carried: '))

value = [int(v) for v in value]
weight = [int(w) for w in weight]


max_value, fractions = fractional_knapsack(value, weight, capacity)


print('The maximum value of items that can be carried:', max_value)
print('The fractions in which the items should be taken:', fractions)


# 60 100 120
# weights: 10 20 30
# maxweight : 50
# maxvalue : 240
```

TEST CASE:

Case 1:

Enter number of items: 3

Enter the values of the 3 item(s) in order: 60 100 120

Enter the positive weights of the 3 item(s) in order: 10 20 30

Enter maximum weight: 50

The maximum value of items that can be carried: 240.0

The fractions in which the items should be taken: [1, 1, 0.6666666666666666]

```
PS C:\Users\HARSH-PC\Desktop\college\AI\EXP_2> .\knapsack.py
Number of items: 3
Enter the values of the 3 item(s) in order: 60 100 120
Enter the positive weights of the 3 item(s) in order: 10 20 30
Enter maximum weight that can be carried: 50
The maximum value of items that can be carried: 240.0
The fractions in which the items should be taken: [1, 1, 0.6666666666666666]
PS C:\Users\HARSH-PC\Desktop\college\AI\EXP_2>
```

Case 2:

Enter number of items: 5

Enter the values of the 5 item(s) in order: 3 5 1 2 4

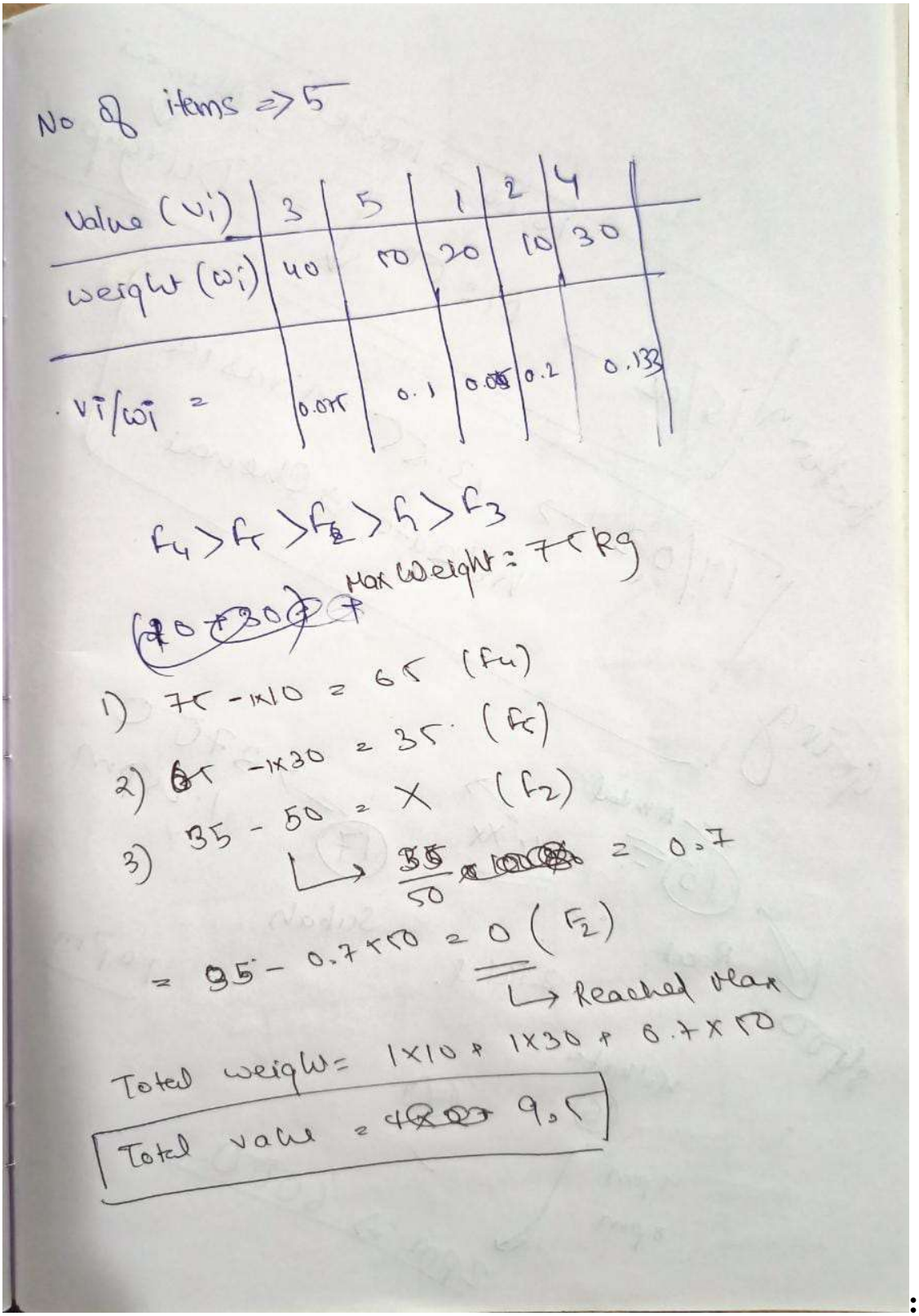
Enter the positive weights of the 5 item(s) in order: 40 50 20 10 30

Enter maximum weight: 75

The maximum value of items that can be carried: 9.5

The fractions in which the items should be taken: [0, 0.7, 0, 1, 1]

Verification



Result: We have successfully solved the Fractional Knapsack problem using Greedy approach in Python and verified the output and test cases.