RA1911042010072

CSBS2

AI LAB EXPERIMENT 3

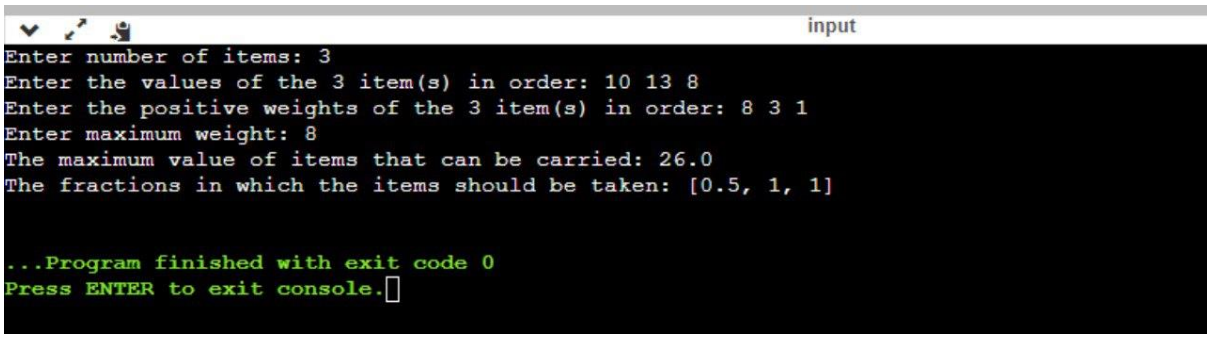
Developing agent programs for real world problems (Knapsack problem using greedy method)

Code –

```
def fractional_knapsack(value, weight, capacity):
    # index = [0, 1, 2, ..., n - 1] for n items
    index = list(range(len(value)))
    # contains ratios of values to weight
    ratio = [v/w for v, w in zip(value, weight)]
    # index is sorted according to value-to-weight ratio in decreasing order
    index.sort(key=lambda i: ratio[i], reverse=True)

    max_value = 0
    fractions = [0]*len(value)
    for i in index:
        if weight[i] <= capacity:
            fractions[i] = 1
            max_value += value[i]
            capacity -= weight[i]
        else:
            fractions[i] = capacity/weight[i]
            max_value += value[i]*capacity/weight[i]
            break

    return max_value, fractions



n = int(input('Enter number of items: '))
value = input('Enter the values of the {} item(s) in order: '
              .format(n)).split()
```

```
value = [int(v) for v in value]
weight = input('Enter the positive weights of the {} item(s) in order: '
               .format(n)).split()
weight = [int(w) for w in weight]
capacity = int(input('Enter maximum 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)
```

Output –



EXPLANATION –

Greedy algorithms are used to solve optimization problems, i.e., find the best solution based upon given criteria. Greedy algorithms implement optimal local selections in the hope that those selections will lead to the best solution. However, the solution to the greedy method is always not optimal. Greedy methods work well for the fractional knapsack problem. However, for the 0/1 knapsack problem, the output is not always optimal . Inconclusion, The greedy method's idea is to calculate the (value/weight) ratio. Sort the ratios in descending order. Choose the first ratio, which is the maximum package. The capacity of the knapsack can contain that package (remain > weight). Every time a package is put into the knapsack, it will also reduce the knapsack's capacity. Return maximum value of items and their fractional amounts. (max_value, fractions) is returned where max_value is the maximum value of items with total weight not more than capacity. fractions is a list where fractions[i] is the fraction that should be taken of item i, where 0 <= i < total number of items. value[i] is the value of item i and weight[i] is the weight of item i for 0 <= i < n where n is the number of items. capacity is the maximum weight.