

Objective: Implementation and analysis of Knapsack problem

Knapsack problem

Given a set of items, each with a weight and a value, determine a subset of items to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

The knapsack problem is in combinatorial optimization problem. It appears as a subproblem in many, more complex mathematical models of real-world problems. One general approach to difficult problems is to identify the most restrictive constraint, ignore the others, solve a knapsack problem, and somehow adjust the solution to satisfy the ignored constraints.

Knapsack Algorithm:

```
Greedy-Fractional-Knapsack (w[1..n], p[1..n], W)
for i = 1 to n
  do x[i] = 0
weight = 0
for i = 1 to n
  if weight + w[i] ≤ W then
    x[i] = 1
    weight = weight + w[i]
  else
    x[i] = (W - weight) / w[i]
    weight = W
    break
return x
```

Code:

```
def fractional_knapsack(value, weight, capacity):
    index = list(range(len(value)))
    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)
    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)

```

The screenshot shows a code editor with the following Python code for a fractional knapsack problem:

```

1 def fractional_knapsack(value, weight, capacity):
2     index = list(range(len(value)))
3     ratio = [v / w for v, w in zip(value, weight)]
4     index.sort(key=lambda i: ratio[i], reverse=True)
5     max_value = 0
6     fractions = [0] * len(value)
7     for i in index:
8         if weight[i] <= capacity:
9             fractions[i] = 1
10            max_value += value[i]
11            capacity -= weight[i]
12        else:
13            fractions[i] = capacity / weight[i]
14            max_value += value[i] * capacity / weight[i]
15            break
16    return max_value, fractions
17
18 n = int(input('Enter number of items: '))
19 value = input('Enter the values of the {} item(s) in order: '.format(n)).split()
20 value = [int(v) for v in value]
21 weight = input('Enter the positive weights of the {} item(s) in order: '.format(n)).split()
22 weight = [int(w) for w in weight]
23 capacity = int(input('Enter maximum weight: '))
24 max_value, fractions = fractional_knapsack(value, weight, capacity)
25 print('The maximum value of items that can be carried:', max_value)
26 print('The fractions in which the items should be taken:', fractions)

```

The IDE interface includes a menu bar (File, Edit, View, etc.), a toolbar with icons for running and debugging, and a status bar at the bottom showing the current file (Knapsack.py), Python version (3.8), and system time (05:40 PM, 14-11-2020).

Output:

Enter number of items: 3

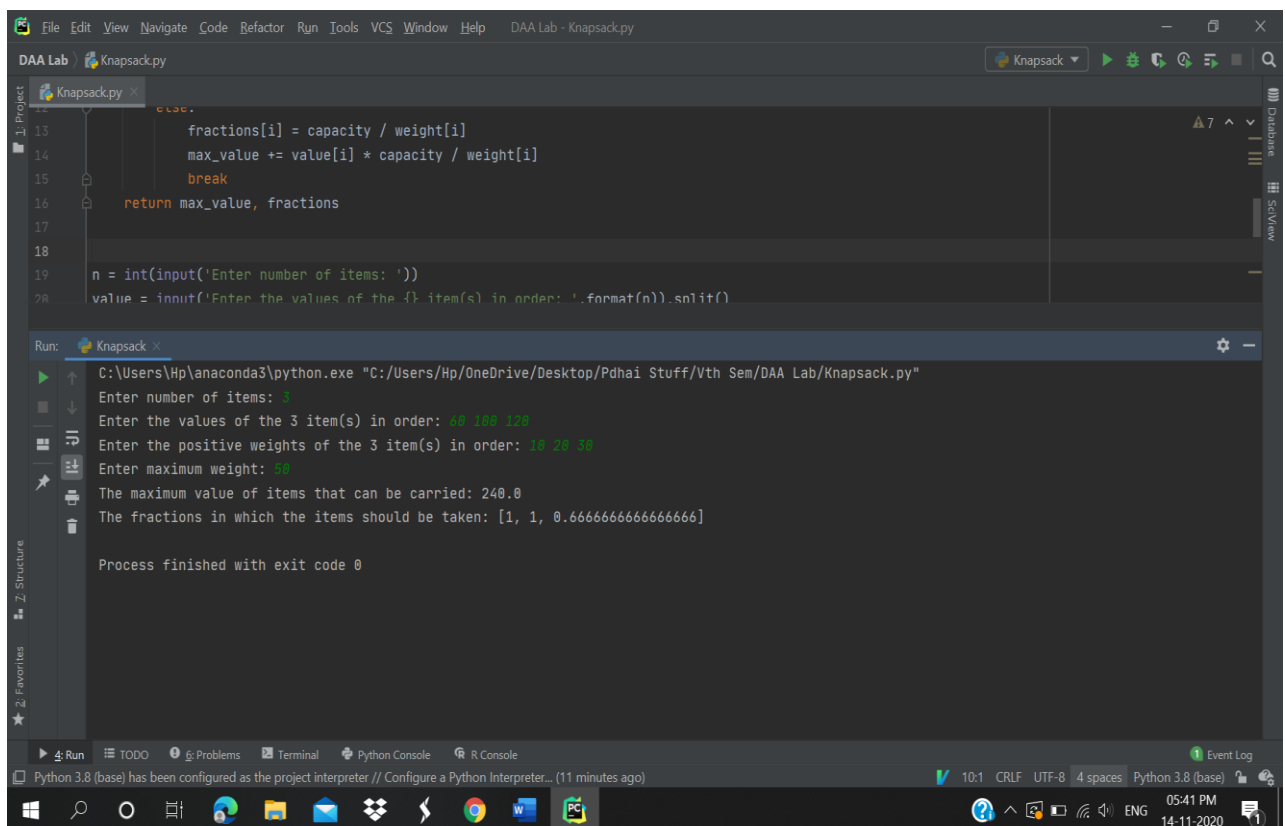
Enter the values of the 3 item(s) in order: 60 100 120

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]



The screenshot shows a code editor with the following Python code for a knapsack problem:

```
13     else:
14         fractions[i] = capacity / weight[i]
15         max_value += value[i] * capacity / weight[i]
16         break
17     return max_value, fractions
18
19 n = int(input('Enter number of items: '))
20 value = input('Enter the values of the {} item(s) in order: '.format(n)).split()
```

The Run console shows the following output:

```
C:\Users\Hp\anaconda3\python.exe "C:/Users/Hp/OneDrive/Desktop/Pdhai Stuff/Vth Sem/DAA Lab/Knapsack.py"
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]
Process finished with exit code 0
```

Time Complexities:

If the provided items are already sorted into a decreasing order of p_i/w_i , then the while loop takes a time in $O(n)$; Therefore, the total time including the sort is in $O(n \log n)$.

Knapsack Sort Applications:

- Finding the least wasteful way to cut raw materials
- portfolio optimization
- Cutting stock problems