

Assignment No: 4

Name: Kedar Kale

Roll No: SCETTYB129

Batch: B12

Title: Write a program for Fractional Knapsack using greedy method.

Algorithm:

```
for i = 1 to n
    do  $x[i] = 0$ 
weight = 0
for i = 1 to n
    if  $\text{weight} + w[i] \leq W$  then
         $x[i] = 1$ 
         $\text{weight} = \text{weight} + w[i]$ 
    else
         $x[i] = (W - \text{weight}) / w[i]$ 
         $\text{weight} = W$ 
        break
return x
```

Time Complexity:

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 $O(n \log n)$.

Analysis:

algorithm FractionalKnapsack(S, W):

Input: Set S of items i with weight w_i and benefit b_i all positive.

Knapsack capacity $W > 0$.

Output: Amount x_i of i that maximizes the total benefit without exceeding the capacity.

for each i in S do

$x_i \leftarrow 0$ { for items not chosen in next phase }

$v_i \leftarrow b_i/w_i$ { the value of item i "per pound" }

$w \leftarrow W$ { remaining capacity in knapsack }

while $w > 0$ do

 remove from S an item of maximal value { greedy choice }

$x_i \leftarrow \min(w_i, w)$ { can't carry more than w more }

$w \leftarrow w - x_i$

Program:

```
#include<iostream>
using namespace std;
void knapsack(int n, float weight[], float profit[], float capacity) {
    float x[20], tp = 0;
    int i, j, u;
    u = capacity;
    cout<<"Weight"<<" "<<"Profit"<<"\n";
    for (i = 0; i < n; i++)
        x[i] = 0.0;

    for (i = 0; i < n; i++) {
        if (weight[i] > u)
            break;
        else {

            cout<<" "<<weight[i]<<"    "<<profit[i];
            cout<<"\n";

            x[i] = 1.0;
            tp = tp + profit[i];
            u = u - weight[i];
        }
    }

    if (i < n)
        x[i] = u / weight[i];

    tp = tp + (x[i] * profit[i]);
    cout<<" "<<(x[i]*weight[i])<<"    "<<(x[i]*profit[i]);

    cout<<"\nMaximum profit is:-"<<tp;
}

int main() {
    float weight[20], profit[20], capacity;
    int num, i, j;
    float ratio[20], temp;

    cout<<"\nEnter the no. of objects:- ";
    cin>>num;

    cout<<"\nEnter the wts and profits of each object:- ";
    for (i = 0; i < num; i++) {
        cin>>weight[i]>>profit[i];
    }

    cout<<"\nEnter the capacity of knapsack:- ";
    cin>>capacity;

    for (i = 0; i < num; i++) {
        ratio[i] = profit[i] / weight[i];
    }
}
```

```

    }

    for (i = 0; i < num; i++) {
        for (j = i + 1; j < num; j++) {
            if (ratio[i] < ratio[j]) {
                temp = ratio[j];
                ratio[j] = ratio[i];
                ratio[i] = temp;

                temp = weight[j];
                weight[j] = weight[i];
                weight[i] = temp;

                temp = profit[j];
                profit[j] = profit[i];
                profit[i] = temp;
            }
        }
    }

    knapsack(num, weight, profit, capacity);
    return(0);
}

```

Input/Output:

```

1 #include<iostream>
2 using namespace std;
3 void knapsack(int n, float weight[], float profit[], float capacity) {
4     float x[20], tp = 0;
5     int i, j, u;
6     u = capacity;
7     cout<<"Weight"<<" "<<"Profit"<<"\n";
8     for (i = 0; i < n; i++)
9         x[i] = 0.0;
10
11     for (i = 0; i < n; i++) {
12         if (weight[i] > u)
13             break;
14         else {
15             cout<<" "<<weight[i]<<" "<<profit[i],
16             cout<<"\n";
17
18             x[i] = 1.0;
19             tp = tp + profit[i];
20             u = u - weight[i];
21         }
22     }
23
24     if (i < n)
25         x[i] = u / weight[i];
26
27     tp = tp + (x[i] * profit[i]);
28     cout<<" "<<(x[i]*weight[i]<<" "<<(x[i]
29
30     cout<<"\nMaximum profit is:-"<<tp;
31 }
32
33 int main() {
34     float weight[20], profit[20], capacity;
35     int num, i, j;
36     float ratio[20], temp;
37
38     cout<<"\nEnter the no. of objects:- ";
39     cin>>num;
40

```

```

student@localhost:~$ g++ napsack.cpp
napsack.cpp: In function 'void knapsack(int, float*, float*, float)':
napsack.cpp:29:23: error: invalid operands of types 'float' and 'float*' to binary 'operator'
    cout<<" "<<(x[i]*weight)<<" "<<(x[i]*profit[i]);
                        ^~~~~~
student@localhost:~$ g++ napsack.cpp
student@localhost:~$ ./a.out
Enter the no. of objects:- 4
Enter the wts and profits of each object:- 10 100
40 280
20 120
24 120
Enter the capacity of knapsack:- 60
Weight Profit
10 100
40 280
10 60
Maximum profit is:-440student@localhost:~$

```

Conclusion: We have successfully executed the Fractional Knapsack Problem using Greedy Approach.