Suraj Iyer
2021300045
SE Comps A
Batch C

DAA Experiment 5

Aim - Fractional Knapsack Problem - Greedy Approach

Details - 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.

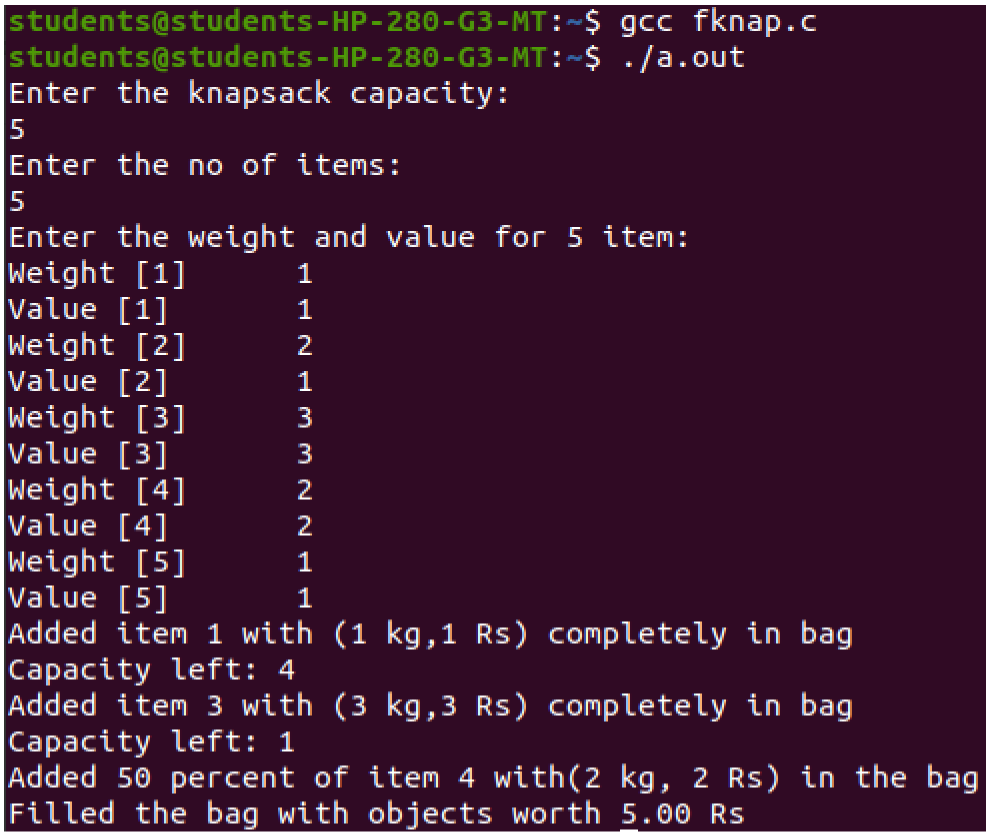
Follow the given steps to solve the problem using the above approach:

- Calculate the ratio(value/weight) for each item.
- Sort all the items in decreasing order of the ratio.
- Initialize res =0, curr_cap = given_cap.
- Do the following for every item "i" in the sorted order:
    - If the weight of the current item is less than or equal to the remaining capacity then add the value of that item into the result
    - Else add the current item as much as we can and break out of the loop.
- Return res.

Code -

```
#include <stdio.h>

void main()
{
        int W,curr_w,maxi,n;
        int used[10];
        int weight[10];
        int val[10];
        float tot_val;
        int i;

        printf("Enter the knapsack capacity:\n");
        scanf("%d",&W);
        printf("Enter the no of items:\n");
        scanf("%d",&n);
        printf("Enter the weight and value for %d item:\n",n);
        for(i=0;i<n;i++)
        {
        printf("Weight [%d]\t",i+1);
        scanf("%d",&weight[i]);
        printf("Value [%d]\t",i+1);
        scanf("%d",&val[i]);
        }

        //printf("Weight\n%d\t Value\n%d\t",weight[i],val[i]);

        for(i=0;i<n;++i)
        {
        used[i]=0; //since no ith item has been used
        }

        curr_w=W;
        while(curr_w>0)
        {
        maxi=-1;//finding the best item to put in bag
        for(i=0;i<n;++i)
        {
                if((used[i]==0)&&((maxi==-1) || ((float)val[i]/weight[i] >
(float)val[maxi]/weight[maxi])))
                {
                maxi=i;
                }
        }
        used[maxi]=1; //item already in the bag
        curr_w-=weight[maxi];//one one item going to be less if put in the bag
        tot_val+=val[maxi];

        if(curr_w>=0)
        {
                printf("Added item %d with (%d kg,%d Rs) completely in
bag\n",maxi+1,weight[maxi],val[maxi]);
                printf("Capacity left: %d\n",curr_w);
        }
        else{
        printf("Added %d percent of item %d with(%d kg, %d Rs) in the
bag\n",(int)((1+(float)curr_w/weight[maxi])*100),maxi+1,weight[maxi],
val[maxi]);
        tot_val -= val[maxi];
        tot_val += (1 + (float)curr_w/weight[maxi]) * val[maxi]; //adding that
percent of value in the bag as left
        }
        }
        printf("Filled the bag with objects worth %.2f Rs\n", tot_val);
}
```

Output -



```
students@students-HP-280-G3-MT:~$ gcc fknap.c
students@students-HP-280-G3-MT:~$ ./a.out
Enter the knapsack capacity:
5
Enter the no of items:
5
Enter the weight and value for 5 item:
Weight [1]      1
Value [1]       1
Weight [2]      2
Value [2]       1
Weight [3]      3
Value [3]       3
Weight [4]      2
Value [4]       2
Weight [5]      1
Value [5]       1
Added item 1 with (1 kg,1 Rs) completely in bag
Capacity left: 4
Added item 3 with (3 kg,3 Rs) completely in bag
Capacity left: 1
Added 50 percent of item 4 with(2 kg, 2 Rs) in the bag
Filled the bag with objects worth 5.00 Rs
```

Conclusion - We have hence found out the maximum value that we can fit in our knapsack following the criteria that the items have a certain weight and value by taking into consideration the value by weight ratio which allows us to prioritize certain items over others. The fact that fractions of certain items can also be added in order to fully fill the knapsack up to the brim is also tackled using the value by weight ratio approach.