Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

Academic Year 2022-23 (Even)

Name of student: DIVYESH KHUNT **SAP ID:60009210116**

Course : Design and Analysis of Algorithms Lab **Course Code:**

Experimentno1

Aim: Implementation of Fractional Knapsack Problem using Greedy Approach.

Theory:

Given the weights and values of **N** items, in the form of **{value, weight}** put these items in aknapsack 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 remainingcapacity 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**.

Department of Computer Science and Engineering (Data Science)

```c
#include<stdio.h>
#include<conio.h>
int main()
{
    int n;
    float temp;
    float total_profit = 0.00;
    printf("Enter the size: ");
    scanf("%d", &n);
    float item[n], profit[n], weight[n];
    float  p_w[n], size_array[n];
    for(int i=0; i<n; i++)
    {
        item[i]=i;
    }
    printf("\nEnter the profit: ");
    for(int i=0; i<n; i++)
    {
        scanf("%f", &profit[i]);
    }
    printf("\nEnter the weight: ");
    for(int i=0; i<n; i++)
    {
        scanf("%f", &weight[i]);
    }
    for(int i=0; i<n; i++)
    {
        p_w[i] = profit[i]/weight[i];
    }
```

```
    float sack_weight;
    printf("\nEnter the sack weight: ");
    scanf("%f", &sack_weight);
    for(int i=0; i<n; i++)
    {
        for(int j=i+1; j<n; j++)
        {
            if(p_w[i]<p_w[j])
            {
                temp=p_w[i];
                p_w[i]=p_w[j];
                p_w[j]=temp;

                temp=item[i];
                item[i]=item[j];
                item[j]=temp;

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

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

            }
        }
    }
    for(int i=0; i<n; i++)
    {
        printf("\n%.2f %.2f %.2f %.2f", item[i], profit[i], weight[i], p_w[i]);
    }
    for(int i=0; i<n; i++)
    {
        if(sack_weight>=weight[i])
        {
            sack_weight = sack_weight - weight[i];
            total_profit = total_profit + profit[i];
            size_array[i] = 1;
        }
        else
        {
            float tempo_size = sack_weight/weight[i];
            sack_weight = sack_weight - weight[i];
            float tempo_profit = tempo_size * profit[i];
            total_profit = total_profit + tempo_profit;
            size_array[i] = tempo_size;
        }
    }
    printf("\n\nTotal profit: %.2f", total_profit);
    printf("\n\nThe size_array is: ");
    for(int i=0; i<n; i++)
    {
        printf("%.2f\t", size_array[i]);
    }
}
```

Output –

```
Enter the size: 4

Enter the profit: 10
5
20
16

Enter the weight: 5
2
4
4

Enter the sack weight: 12

2.00 20.00 4.00 5.00
3.00 16.00 4.00 4.00
1.00 5.00 2.00 2.50
0.00 10.00 5.00 2.00

Total profit: 45.00

The size_array is: 1.00 1.00    1.00    0.40

...Program finished with exit code 0
Press ENTER to exit console.
```

Time Complexity: O(N * log N)

Example

Complexity :

Time Complexity: **O(N * log N)**
Auxiliary Space: **O(N)**

Conclusion:

Thus the code of ACTIVITY SELECTION PROBLEM was successfully implemented