



**Jawahar Education Societys Annasaheb Chudaman Patil College of
Engineering, Kharghar, Navi Mumbai**

NAME: PRIYUSH BHIMRAO KHOBRADE

PRN NO: 211112018

Roll No: 52

SUBJECT: Analysis of Algorithms Lab

Knapsack Problem using Greedy method

EXPERMINT: 04

PAGE NO.:

DATE: / / 20

Experiment No- 04

- Aim :- write a C-Program for knapsack problem using greedy method.
- Objectives :- To implement and analyze complexity of knapsack problem.
- Outcomes :- Student will be able to complete the complexity of knapsack.
- Hardware / Software Required :- 'C' compiler.

Problem Definition :-

There are no. of objects and knapsack or bag. Each object i has weight w_i and a knapsack with capacity m . if a fraction x_i , $0 \leq x_i \leq 1$ of object i is placed into knapsack, then the profit $x_i * p_i$ is earned. The objective is to obtain a filling of knapsack that maximizes the object total profit earned. The total weight of all chosen object must be M .

The program can be started as:

~~maximum~~
Maximize $\sum_{i=1}^n x_i * p_i$

Subject to $\sum w_i * x_i \leq M$

1

Teachers Signature _____

Knapsack Problem using Greedy method

PAGE NO.:

DATE.: / / 20

• Theory :-

* Optimization problem :- These can be many possible solution solutions. Each solution has value and we wish to find a solution with the optimal (minimum or maximum) value.

* Greedy algorithm :- is an algorithmic technique to solve optimization problems it always make the choice that looks best at the moment. Then it make a locally optimal choice in the hope that this choice will lead to a globally optimal solution.

• Algorithm :-

Algorithm Knapsack (m, n, p, w);

// Description :-

// Input: profit & weight of n object in decreasing order of $p[i]/w[i]$
m is size of knapsack.

// output :- array x of 1 and 0 which shows object gives maximum profit.

for $i = 1$ to n do

$x[i] = 0$

$u = m$

for $j = 1$ to n do

}

if $(u - w[i] \geq 0)$

}

$u = u - w[i]$

$x[i] = 1, 0$

}

}

2

Teachers Signature _____

Knapsack Problem using Greedy method

PAGE NO.:

DATE.: / / 20

Otherwise break.

}

// end of knapsack.

• Analysis :-

$$T(n) := \sum_{i=1}^n 1 + \sum_{i=1}^n 1$$

$$= n + n = 2n$$

$$= O(n)$$

• Conclusion :- Thus the complexity of knapsack problem is $O(n)$.

3

Teachers Signature _____

Knapsack Problem using Greedy method

Input:

```
1 #include<stdio.h>
2 void kp(int n, float wt[], float pro[], float capacity);
3 void main()
4 {
5     int i,j,n;
6     float wt[20],pro[20],ratio[20],capacity;
7     float temp;
8     printf("Enter no. of object:");
9     scanf("%d",&n);
10    printf("Enter the capacity:");
11    scanf("%f",&capacity);
12    printf("Enter the weights\n");
13    for(i=0;i<n;i++)
14    {
15        printf("\n\WT %d:",i+1);
16        scanf("%f",&wt[i]);
17    }
18    printf("Enter the profit\n");
19    for(i=0;i<n;i++)
20    {
21        printf("\npro %d:",i+1);
22        scanf("%f",&pro[i]);
23    }
24    printf("CAL ratios\n");
25    for(i=0;i<n;i++)
26    {
27        ratio[i]=pro[i]/wt[i];
28        printf("%f ",ratio[i]);
29    }
30    printf("\nSorting\n");
31    for(i=0;i<n;i++)
32    {
33        for(j=i+1;j<n;j++)
34        {
35            if(ratio[i]<ratio[j])
36            {
37                temp=wt[i];
38                wt[i]=wt[j];
39                wt[j]=temp;
40                temp=pro[i];
41                pro[i]=pro[j];
42                pro[j]=temp;
43                temp=ratio[i];
44                ratio[i]=ratio[j];
45                ratio[j]=temp;
46            }
47        }
48    }
49    for(i=0;i<n;i++)
50    printf("%f ",pro[i]);
51    for(i=0;i<n;i++)
52    printf("%f ",wt[i]);
53
54    kp(n,wt,pro,capacity);
55 }
56
57 void kp(int n,float wt[],float pro[],float capacity)
58 {
59     int i,u;
60     float x[20];
61     float p=0.0;
62     for(i=0;i<n;i++)
63     {
64         x[i]=0.0;
65     }
66     u=capacity;
67     for(i=0;i<n;i++)
68     {
69         if(wt[i]<=u)
70         {
71             x[i]=1.0;
72             u=u-wt[i];
73         }
74         else
75         {
76             break;
77         }
78     }
79     x[i]=u/wt[i];
80     printf("\nSolution vector:");
81     for(i=0;i<n;i++)
82     {
83         printf("%f ",x[i]);
84     }
85     for(i=0;i<n;i++)
86     p=p+(x[i]*pro[i]);
87     printf("\nMaximum Profit is:%f",p);
88 }
```

Knapsack Problem using Greedy method

Output:

```
C:\Users\Rupesh\Documents\OS\AOAEX04\bin\Debug\AOAEX04.exe
WT 1:10
WT 2:20
WT 3:30
WT 4:40
WT 5:50
Enter the profit
pro 1:20
pro 2:30
pro 3:66
pro 4:40
pro 5:60
CAL ratios
2.000000 1.500000 2.200000 1.000000 1.200000
Sorting
66.000000 20.000000 30.000000 60.000000 40.000000 30.000000 10.000000 20.000000 50.000000 40.000000
Solution vector:1.000000 1.000000 1.000000 0.800000 0.000000
Maximum Profit is:164.000000
Process returned 29 (0x1D)   execution time : 90.598 s
Press any key to continue.
```

Conclusion: Thus the Complexity of Knapsack problem is $O(n)$.