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

NAME: PRIYUSH BHIMRAO KHOBRAGADE

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 Requisite :- 'C' compiler.
- Problem Definition :-

There are no. of objects and knapsack or bag. Each object i has weight w, and a knapsack with capacity m. if a fraction x_i, $0 <= x <= 1$ of object i is placed into knaspack, then the profit $X_i * P_i$ is ended. The objective is to obtain a filling of knapsack that maximum the object total profit earned. The total weight of all chosen object must be M.

The problem can be stated as:

Maximize $\sum_{i<i<n} x_i * P_i$

Subject to $\sum w_i * x_i < M$

Teachers Signature _____

Knapsack Problem using Greedy method

PAGE NO.:
DATE.: / / 20

• Theory :-

* Optimization problem :- There 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,pw);
// Description:-
// Input:- Profit & weight on a object in decreasing 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 i=1 to n do
  {
   if(u-w[i]>=0)
   {
   u= u-w[i]
   x[i]=1.0
  }
}
```

2

Teachers Signature _____

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 $O(n)$.

Teachers Signature _____

Knapsack Problem using Greedy method

Input:

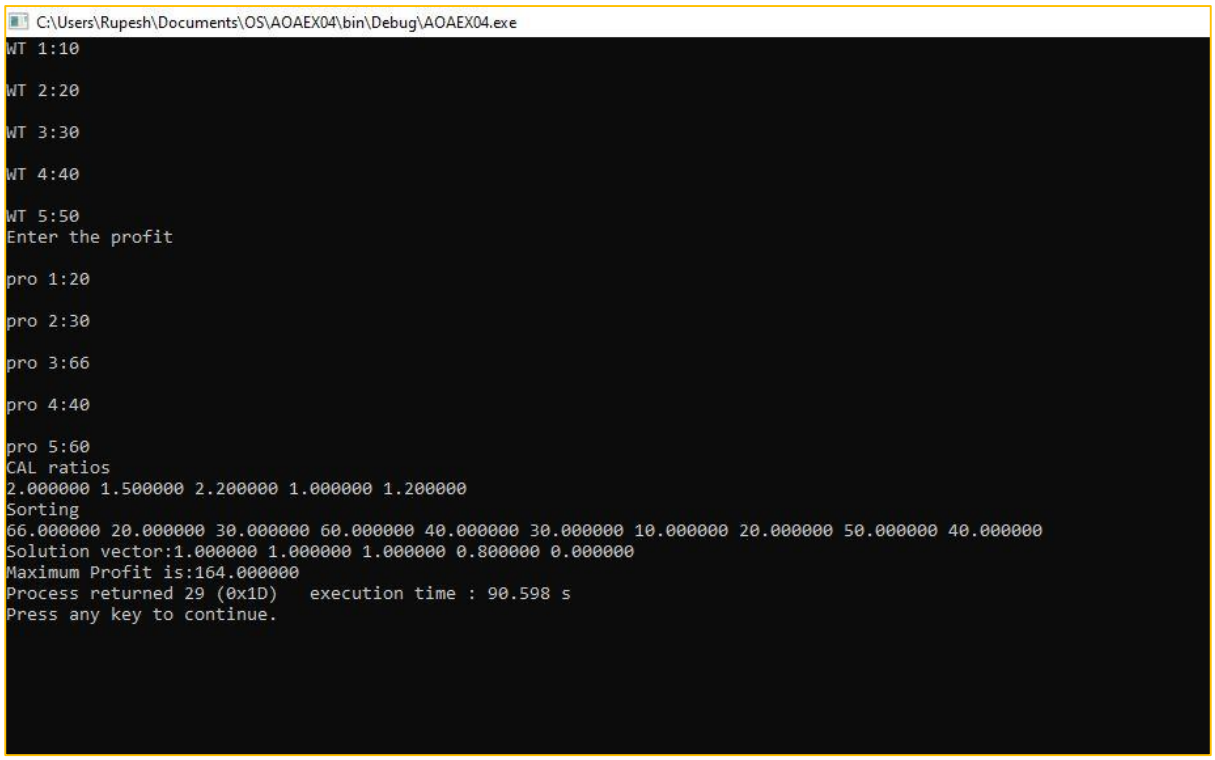
```
#include<stdio.h>
void kp(int n, float wt[], float pro[], float capacity);
void main()
{
int i,j,n;
float wt[20],pro[20],ratio[20],capacity;
float temp;
printf("Enter no. of object:");
scanf("%d",&n);
printf("Enter the capacity:");
scanf("%f",&capacity);
printf("Enter the weights\n");
for(i=0;i<n;i++)
{
printf("\nWT %d:",i+1);
scanf("%f",&wt[i]);
}
printf("Enter the profit\n");
for(i=0;i<n;i++)
{
printf("\npro %d:",i+1);
scanf("%f",&pro[i]);
}
printf("CAL ratios\n");
for(i=0;i<n;i++)
{
ratio[i]=pro[i]/wt[i];
printf("%f ",ratio[i]);
}
printf("\nSorting\n");
for(i=0;i<n;i++)
{
for(j=i+1;j<n;j++)
{
if(ratio[i]<ratio[j])
{
temp=wt[i];
wt[i]=wt[j];
wt[j]=temp;
temp=pro[i];
pro[i]=pro[j];
pro[j]=temp;
temp=ratio[i];
ratio[i]=ratio[j];
ratio[j]=temp;
}
}
}
for(i=0;i<n;i++)
printf("%f ",pro[i]);
for(i=0;i<n;i++)
printf("%f ",wt[i]);

kp(n,wt,pro,capacity);
}

void kp(int n,float wt[],float pro[],float capacity)
{
int i,u;
float x[20];
float p=0.0;
for(i=0;i<n;i++)
{
x[i]=0.0;
}
u=capacity;
for(i=0;i<n;i++)
{
if(wt[i]<=u)
{
x[i]=1.0;
u=u-wt[i];
}
else
{
break;
}
}
x[i]=u/wt[i];
printf("\nSolution vector:");
for(i=0;i<n;i++)
{
printf("%f ",x[i]);
}
for(i=0;i<n;i++)
p=p+(x[i]*pro[i]);
printf("\nMaximum Profit is:%f",p);
}
```

Knapsack Problem using Greedy method

Output:



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