PROGRAM 6

**6)**

Implement in Java, the 0/1 Knapsack problem using

(a) Dynamic Programming method

(b) Greedy method.

package labprograms;

import java.util.Scanner;

public class p6 {

static int[] w=new int[50];

static int[] p=new int[50];

static int[] x=new int[50];

static int[] t=new int[50];

static double maxprofit;

static int n,m,i,j;

static void dk(int n,int w[],int p[],int m) {

int[][] v=new int[n+1][m+1];

for(j=0;j<=m;j++)

v[0][j]=0;

for(i=0;i<=n;i++)

v[i][0]=0;

for(i=1;i<=n;i++)

for(j=1;j<=m;j++)

if(j<w[i])

v[i][j]=v[i-1][j];

else

v[i][j]=max(v[i-1][j],v[i-1][j-w[i]]+p[i]);

System.out.println("Solution Table is : ");

for(i=0;i<=n;i++) {

for(j=0;j<=m;j++)

System.out.print(v[i][j]+"\t");

System.out.print("\n");

}

System.out.println("The optimal solution for dynamic method is -->"+v[n][m]);

i=n;

j=m;

while((i!=0)&&(j!=0))

{

if(v[i][j]!=v[i-1][j])

{

x[i]=1;

j=j-w[i];

}

i=i-1;

}

System.out.println("The solution vector for Dynamic method is : ");

for(i=1;i<=n;i++)

System.out.print(x[i]+"\t");

System.out.println("\n");

}

static int max(int a,int b)

{

return (a>b)?a:b;

}

static void gk(int n,int w[],int p[],int m) {

int rc=m;

bs(n,w,p,t);

for(i=1;i<=n;i++)

{

if(w[t[i]]>rc)

continue;

x[t[i]]=1;

rc-=w[t[i]];

maxprofit+=p[t[i]];

}

System.out.println("Optimal solution for Greedy Method --> "+maxprofit);

System.out.println("The solution vector for Greedy Method : ");

for(int i=1;i<=n;i++)

System.out.print(x[i]+"\t");

}

static void bs(int n,int w[],int p[],int t[]) {

int temp;

for(i=1;i<=n;i++)

t[i]=1;

for(i=1;i<n;i++)

for(j=1;j<=n;j++)

if((double)p[t[j]]/w[t[j]]<(double)p[t[j+1]]/w[t[j+1]])

{

temp=t[j];

t[j]=t[j+1];

t[j+1]=temp;

}

}

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

System.out.print("Enter the number of objects : ");

n=sc.nextInt();

System.out.println("Enter the objects weights : ");

for(i=1;i<=n;i++)

w[i]=sc.nextInt();

System.out.println("Enter the objects profits : ");

for(i=1;i<=n;i++)

p[i]=sc.nextInt();

System.out.print("Enter the maximum capacity : ");

m=sc.nextInt();

dk(n,w,p,m);

for(i=1;i<=n;i++)

x[i]=0;

gk(n,w,p,m);

sc.close();

} }