Knapsack-Dynamic_programming

Implement in Java, the 0/1 Knapsack problem using Dynamic Programming method

Input Format

5 20 3 4 5 6 8 2 3 4 5 6

Constraints

--

Output Format

Optimal Solution: 26 The objects picked up into knapsack are: 5 4 3 2 1

Sample Input 0

```
5
20
3
4
5
6
8
2
3
4
5
6
```

Sample Output 0

```
Optimal Solution: 26
The objects picked up into knapsack are:
5 4 3 2 1
```


Java 7

```java
//224G1A0553
import java.util.Scanner;
class DKnapsack {
 int c,n,p[],w[],v[][];
 public DKnapsack(int n,int c,int[] p,int[] w)
 {
 super();
 this.n=n;
 this.c=c;
 this.w=w;
```

```
this.p=p;
this.v=new int[n+1][c+1];
}
void compute()
{
for(int i=0;i<=n;++i){
for(int j=0;j<=c;++j){
if(i==0||j==0){
v[i][j]=0;
}
else if(j-w[i]>=0)
{
v[i][j]=max(v[i-1][j],p[i]+v[i-1][j-w[i]]);
}
else if(j-w[i]<0)
{
v[i][j]=v[i-1][j];
}}}
System.out.println("Optimal Solution: "+v[n][c]);
traceback();
}
void traceback(){
System.out.println("The objects picked up into knapsack are:");
int i=n;
int j=c;
while(i>0)
{
if(v[i][j]!=v[i-1][j])
{
System.out.print(i+" ");
j=j-w[i];
i--;
}
else {
i--;
}}}
private int max(int i,int j){
if(i>j)return i;
else return j;
}}
public class KpDynamic{
public static void main(String[] args){
int c,n;
Scanner input=new Scanner(System.in);
//System.out.println("Enter number of objects");
n=input.nextInt();
int[] p=new int[n+1];
int[] w=new int[n+1];
int i;
//System.out.println("Enter capacity of Knapsack");
c=input.nextInt();
//System.out.println("Enter profit for each "+n+" objects");
for(i=1;i<=n;i++)
p[i]=input.nextInt();
//System.out.println("Enter weight for each "+n+" objects");
for(i=1;i<=n;i++)
w[i]=input.nextInt();
DKnapsack dk=new DKnapsack(n,c,p,w);
dk.compute();
}}
```

Line: 70 Col: 3

Upload Code as File Test against custom input Run Code Submit Code

Testcase 0 ✔

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
5
20
3
4
5
6
8
2
3
4
5
6
```

Your Output (stdout)

```
Optimal Solution: 26
The objects picked up into knapsack are:
5 4 3 2 1
```

Expected Output

```
Optimal Solution: 26
The objects picked up into knapsack are:
5 4 3 2 1
```