

DESIGN ANALYSIS AND ALGORITHMS

LAB ASSIGNMENT-8

NAME: PRATHAPANI SATWIKA

REG.NO.: 20BCD7160

Write a program that uses backtracking algorithm to solve the 0/1 knapsack problem.

CODE :

```
package Lab8;
import java.util.*;
public class KnapSackUsingBacktracking {
    static int max(int a, int b)
    {
        return (a > b) ? a : b;
    }
    static int knapSack(int cap, int weight[], int price[], int n)
    {
        if (n == 0 || cap == 0)
            return 0;
        if (weight[n - 1] > cap)
            return knapSack(cap, weight, price, n - 1);
        else
            return max(price[n - 1] + knapSack(cap - weight[n - 1], weight, price, n
- 1), knapSack(cap, weight, price, n - 1));
    }
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Please enter the no.of Items in knapSack : ");
        int n = sc.nextInt();
        int [] weight = new int[n];
        int [] price = new int[n];
        System.out.println("Please enter the costs of each item one by one : ");
        for (int i = 0; i < n; i++)
        {
            price[i] = sc.nextInt();
        }
        System.out.println("Please enter the weight of each item one after another : ");
        for (int i = 0; i < n; i++)
        {
            weight[i] = sc.nextInt();
        }
        System.out.println("Please enter the capacity of Knapsack : ");
        int cap = sc.nextInt();
        System.out.println("Our problem looks like :");
    }
}
```

```

        System.out.println(

            "| Item | Weight | Price |"

        );
        for(int i=0;i<n;i++)
        {
            System.out.println("|   "+i+"   |       "+weight[i]+"   |       "+price[i]+"

");
        }
        System.out.println("So,the maximum possible value can be put into knapsack is

: ");
        int k = knapSack(cap, weight, price, n);
        System.out.println(k);
    }
}

```

```

1 package Lab8;
2 import java.util.*;
3 public class KnapSackUsingBacktracking {
4     static int max(int a, int b)
5     {
6         return (a > b) ? a : b;
7     }
8     static int knapSack(int cap, int weight[], int price[], int n)
9     {
10         if (n == 0 || cap == 0)
11             return 0;
12         if (weight[n - 1] > cap)
13             return knapSack(cap, weight, price, n - 1);
14         else
15             return max(price[n - 1] + knapSack(cap - weight[n - 1], weight, price, n - 1),
16                 knapSack(cap, weight, price, n - 1));
17     }
18     public static void main(String args[])
19     {
20         Scanner sc = new Scanner(System.in);
21         System.out.println("Please enter the no.of Items in knapSack : ");
22         int n = sc.nextInt();
23         int [] weight = new int[n];
24         int [] price = new int[n];
25         System.out.println("Please enter the costs of each item one by one : ");
26         for (int i = 0; i < n; i++)
27         {
28             price[i] = sc.nextInt();
29         }
30         System.out.println("Please enter the weight of each item one after another : ");
31         for (int i = 0; i < n; i++)
32         {
33             weight[i] = sc.nextInt();
34         }
35         System.out.println("Please enter the capacity of Knapsack : ");
36         int cap = sc.nextInt();
37         System.out.println("Our problem looks like :");
38         System.out.println(
39
40             "| Item | Weight | Price |"
41
42         );
43         for(int i=0;i<n;i++)
44         {
45             // ...

```

```

44         System.out.println("|   "+i+"   |       "+weight[i]+"   |       "+price[i]+"   |");
45     }
46     System.out.println("So,the maximum possible value can be put into knapsack is : ");
47     int k = knapSack(cap, weight, price, n);
48     System.out.println(k);
49 }
50 }

```

OUTPUT :

```

Please enter the no.of Items in knapSack :
5
Please enter the costs of each item one by one :
30
45
58
60
35
Please enter the weight of each item one after another :
15
22
35
40
10
Please enter the capacity of Knapsack :
80
Our problem looks like :
| Item | Weight | Price |
|  0   |    15  |    30  |
|  1   |    22  |    45  |
|  2   |    35  |    58  |
|  3   |    40  |    60  |
|  4   |    10  |    35  |
So,the maximum possible value can be put into knapsack is :
140

```