43159

ASSIGNMENT-7

TITLE: 0/1 KNAPSACK

PROBLEM STATEMENT: Write a program to solve 0/1 knapsack problem using DP.

THEORY:

- KNAPSACK PROBLEM:

- Knapsack is a problem in computational optimization.

Given a set of items, each with mass & a value, determine the no. of each item to include in a collection so that total weight is less than or equal to a given limit & the total value is as large as possible.

- It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack & must fill it with most valuable items

- ALGORITHM:

i. Let i be the highest numbered item in an optimal solution S for W $.

ii. S = S - {i} is an optimal sol^n for W-Wi $ & the value to solution S is Vi + value(sub-problem)

iii. We can express the fact in the following formula define c[i, w] to be solution for items 1, 2, ..., & the max. weight w

4359

Page No.:
Date:

Algorithm takes the following i/p:

i. Max weight W
ii. No. of items n
iii. Two sequences

$V = \langle v_1, v_2, \ldots, v_n \rangle$

$W = \langle w_1, w_2, \ldots, w_n \rangle$

ALGORITHM:

```
for w=0 to W do
    c[0, w] = 0
for i: 1 to n do
    c[i, 0] = 0

for w = 1 to W do
    if wi ≤ w then
        if vi + c[i-1, w-wi] then
            c[i, w] = vi + c[i-1, w-wi]
        else
            c[i, w] = c[i-1, w]

    else
        c[i, w] = c[i-1, w]
```

Q.1. W = {1, 2, 3}
V = {10, 15, 40}
max W = 6

for 0 weight ⇒ all values = 0

DP[1][1] = max{ DP[0][1], vi + DP[0][1-1] }
= max{ 0, 10 + 0 }
= 10

43159

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	10	10	10	10	10	10
2	0						
3	0						

$\therefore DP[1][0 \ldots 6] = 10$

Now,

$DP[2][2] = \max\{10, 15 + 0\} = 15$

also $DP[2][3] = \max\{DP[2][3], 15 + DP[1][1]\}$

$= \max\{10, 15 + 10\} = 25$

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	10	10	10	10	10	10
2	0	10	15	25	25	25	25
3	0	10	15	40	50	55	65

Hence, max. possible value = 65

Q.2: $W = \{1, 2, 5, 6, 7\}$

$V = \{1, 6, 13, 22, 28\}$

$DP[2][2] = \max\{DP[1][2], 5 + DP[1][2]\}$

$= \max\{1, 6 + 0\} = 6$

$DP[2][3] = 7$

43159

Page No.:
Date:

	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	6	7	7	7	7	7	7	7	7	7
5	0	1	6	7	7	18	19	24	25	25	25	25
6	0	1	6	7	7	18	22	24	29	28	29	29
7	0	1	6	7	7	18	22	28	29	34	35	40
7	0	1	6	7								

Hence max. possible value = 40

CONCLUSION :

Hence, I have learnt about 0/1 knapsack & implemented the same.