Dynamic Programming

Helps to solve complex problem in divide and conquer manner.

It divides the problem into smaller subproblem, solves each of those subproblems exactly ones and stores its solution for later use.

Difference between greedy technique & dynamic programming approach.

Greedy Technique	Dynamic Programming
(1) Explore few possibility	(1) Explore All possibility
(2) Less time	(2) More time
(3) Sometimes give incorrect results.	(3) Always give correct answer.

Applications of Dynamic Programming

(1) Fibonacci Series
(2) Longest Common Subsequence (LCS)
(3) 0/1 Knapsack.
(4) Matrix chain Multiplication
(5) Sum of subset (subset-sum)
(6) All pair shortest path
(7) Optimal cost Binary Search Tree.
(8) Multi-stage Graph.

0/1 Knapsack Problem

- Items cannot be broken.
- It may give incorrect result when resolved using greedy approach.

0/1 KS(m,n) : The maximum profit we will get 0/1 Knapsack problem, where capacity is m and objects are n.

Recurrence Relation:

$$0/1\,KS(m,n) = \begin{cases} 0 & \text{if } m==0 \,||\, n==0 \\ 0/1\,KS(m, n-1) & \text{if } w[n] > m \\ \max\begin{bmatrix} 0/1\,KS(m-w[n], n-1) + P[n] \\ 0/1\,KS(m, n-1) \end{bmatrix} & \text{if } w[n] \leq m \end{cases}$$

$n=3$ $m=10$

objects :	ob_1	ob_2	ob_3
Profits :	80	54	36
weights :	8	6	4
$P/W =$	10	9	9

$\therefore$ Greedy method = 80 ($\because$ Greedy takes max^m profit for 1 unit)

but Actual = 90

$\therefore$ 0/1 Knapsack problem will give wrong answer if we apply Greedy technique.

Manual method

0	0	0	$\Rightarrow$ 0
0	0	1	$\Rightarrow$ 36
0	1	0	$\Rightarrow$ 54
1	0	0	$\Rightarrow$ 80
0	1	1	$\Rightarrow$ 90
1	0	1	$\Rightarrow$ X
1	1	0	$\Rightarrow$ X
1	1	1	$\Rightarrow$ X

max = 90

```
0/1KS (m,n)
{
  if (m==0 || n==0)
       return (0);
  else {
       if (w[n]>m)
            return (0/1KS (m,n-1));
       else {
            a = 0/1KS (m-w[n], n-1) + P[n];
            b = 0/1KS (m,n-1);
            return (max (a,b));
       }
  }
}
```

Ex: $n=5$ $m=22$

Objects :	ob1	ob2	ob3	ob4	ob5
Profit :	25	75	45	85	35
weight :	4	7	3	9	3

0/1 KS(22, 5) $\Rightarrow c$

0/1 KS(19, 4) 0/1 KS(22, 4) $\Rightarrow 2c$

6 levels $2^2c \Leftarrow$ 0/1 KS(10, 3) 0/1 KS(19, 3) 0/1 KS(13, 3) 0/1 KS(22, 3)

$2^3c \Leftarrow$ (7,2) (10,2) (16,2) (19,2) (10,2) (13,2) (19,2) (22,2)

(0,1) (7,1) (3,1) (10,1) . . . .

$$\therefore T.C = c + 2c + 2c^2 + 2^3c + \cdots + 2^n c$$

$$= c[1 + 2 + 2^2 + 2^3 + \cdots + 2^n]$$

increasing G.P series

$$= c\left(\frac{2^n - 1}{2 - 1}\right)$$

$$= O(2^n)$$

Space complexity $=$ i/p $+$ extra
$= n + n$
$= O(n)$

This is without Dynamic Programming.

In the recursive tree, some function calls are repeating, therefore we will go to dynamic programming approach.

How many distinct function call are there in 0/1 KS.

0/1 KS (m, n)

m	n
22	5
21	4
20	
19	3
18	2
⋮	
1	1
0	0

∴ $(m+1)(n+1)$ func. calls

∴ $T.C = O(mn)$

∴ Space complexity = ~~ae~~ i/p + extra

$$= n + \underbrace{n}_{\text{stack}} + \underbrace{mn}_{\text{table}}$$

$$= O(mn)$$

Q weights = {1, 3, 4, 5} Profits = {1, 4, 5, 7}

$m = 7$ $n = 4$

P_i	w_i		0	1	2	3	4	5	6	7
		0	0	0	0	0	0	0	0	0
1	1	1	0	1	1	1	1	1	1	1
4	3	2	0	1	1	4	5	5	5	5
5	4	3	0	1	1	4	5	6	6	9
7	5	4	0	1	1	4	5	7	8	9

profit

x_1	x_2	x_3	x_4
0	1	1	0

$9 - 5 = 4$

Q $n = 4$ $\quad m = 8$

$P = \{1, 2, 5, 6\}$

$w = \{2, 3, 4, 5\}$

tabular method

P_i	w_i	V	0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0	0	1	2	5	5	6	7	7
6	5	4	0	0	1	2	5	6	6	7	8

Profit

$$V[i, w] = \max \begin{cases} V[i-1, w] \\ V[i-1, w - w_i] + P_i \end{cases}$$

$$V[4,1] = \max \begin{cases} V[3,1] \\ V[3, 1-5] + 6 \end{cases}$$

$$= \max \begin{cases} 0 \\ V[3,-4] + 6 \end{cases}$$

undefined

$$V[4,5] = \max \begin{cases} V[3,5] \\ V[3, 5-5] + 6 \end{cases}$$

$$= \max \begin{cases} 5 \\ 0+6 \end{cases}$$

$$= 6$$

Date:

$$v[4,6] = \max\begin{cases} v[3,6] \\ v[3,6-5]+6 \end{cases}$$

$$= \max\begin{cases} 6 \\ 0+6 \end{cases}$$

$$= 6$$

$$v[4,7] = \max\begin{cases} v[3,7] \\ v[3,7-5]+6 \end{cases}$$

$$= \max\begin{cases} 7 \\ 1+6 \end{cases}$$

$$= 7$$

$$v[4,8] = \max\begin{cases} v[3,8] \\ v[3,8-5]+6 \end{cases}$$

$$= \max\begin{cases} 7 \\ 2+6 = 8 \end{cases}$$

$$= 8$$

$\therefore$ solution is (sequence of decision)

x_1	x_2	x_3	x_4
0	1	0	1

$8-6=2$

Set method

$m=8 \quad n=4$

$P=\{1,2,5,6\}$ $\quad (P,w)$
$w=\{2,3,4,5\}$

$S^0=\{(0,0)\}$

$S_1^0=\{(1,2)\}$

$S^1=\{(0,0),(1,2)\}$

$S_1^1=\{(2,3),(3,5)\}$

$S^2=\{(0,0),(1,2),(2,3),(3,5)\}$

Purging Rule
(Dominance Rule)

$S_1^2=\{(5,4),(6,6),(7,7),\cancel{(8,9)}\}$

$S^3=\{(0,0),(1,2),(2,3),\cancel{(3,5)},(5,4),(6,6),(7,7)\}$

$S_1^3=\{(6,5),(7,7),(8,8),\cancel{(11,9)},\cancel{(12,11)},\cancel{(13,12)}\}$

$S^4=\{(0,0),(1,2),(2,3),(5,4),\cancel{(6,6)},(6,5),(7,7),(8,8)\}$

(1) $(8,8) \in S^4$
but $(8,8) \notin S^3$ $\quad \therefore x_4=1$

$(8-6,\ 8-5) = (2,3)$

(2) $(2,3) \in S^3$
and $(2,3) \in S^2$ $\quad \therefore x_3=0$

(3) $(2,3) \in S^2$
but $(2,3) \notin S^1$ $\quad \therefore x_2=1$
$(2-2,\ 3-3)=(0,0)$

(4) $(0,0) \in S^1$ and $(0,0) \in S^0$

$\therefore x_1 = 0$

$\therefore$ Solution is $(0,1,0,1)$

Q $n=3$ $m=6$

$P = \{1,2,5\}$

$w = \{2,3,4\}$

P_i	w_i		0	1	2	3	4	5	6
		0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3
5	4	3	0	0	1	2	5	5	6

Solution is x_1 x_2 x_3

1 0 1 $6-5=1$

Q weights $= \{3,4,6,5\}$ Profits $= \{2,3,1,4\}$

$m=8$ $n=4$.

P_i	w_i		0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
2	3	1	0	0	0	2	2	2	2	2	2
3	4	2	0	0	0	2	3	3	3	5	5
4	5	3	0	0	2	2	3	4	4	5	6
1	6	4	0	0	2	2	3	4	4	5	6

0/1 Knapsack (DP)

Q Weights = { 1, 3, 4, 5, 2 }

Profits = { 1, 4, 5, 7, 4 } $m = 9$ $n = 5$

Soln: Sort items by weight.

$w_i = \{1, 2, 3, 4, 5\}$

$P_i = \{1, 4, 4, 5, 7\}$

P_i	w_i		0	1	2	3	4	5	6	7	8	9
		0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	1	1	1	1	1	1	1	1
4	2	2	0	1	4	5	5	5	5	5	5	5
4	3	3	0	1	4	5	5	8	9	9	9	9
5	4	4	0	1	4	5	5	8	9	10	10	13
7	5	5	0	1	4	5	5	8	9	11	12	13

Profit

x_1	x_2	x_3	x_4	x_5
0	1	1	1	0

$13 - 5 = 8$

$8 - 4 = 4$

$4 - 4 = 0$