0/1 Knapsack

Problem statement

We have a knapsack / bag which we have to fill with objects. If the object is represented with 0 which means object is absent. And if the object is represented with 1 which means object is present.

The knapsack problem which we solve using greedy approach is a fractional knapsack. (profit/weight ratio)
~~This~~
The 0/1 knapsack problem solved using dynamic programming ~~is~~ ~~it~~ has different approach.

Either the full obj is present or absent. No fraction is considered.

Example

object	ob1	ob2	ob3
weight	2	4	8
profit	20	25	60

Knapsack capacity (M) = 12

Our target is to fill the knapsack with the objects to get the maximum profit.

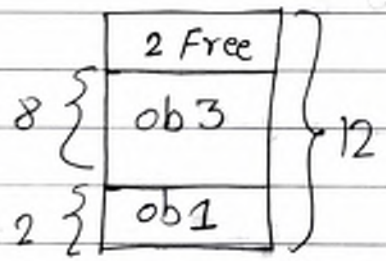
If We first calculate $\frac{\text{profit}}{\text{weight}}$ ratio for each object.

Object	obj1	ob2	ob 3
$\frac{\text{profit}}{\text{weight}}$ ratio	10	6.2	7.5

Highest P/w ratio is for obj1 with weight 2

Profit = 20

Next highest P/w ratio is for obj3 with weight 8



Profit = 20 + 60 = 80

As this is 0/1 knapsack we can not seled the objects in fractional part.

But through this approach we can not achieve the optimal solution.

To solve the 0/1 knapsack problem we ~~regu~~ apply brute-force technique / method.

As per brute-force technique, if if we have n objects as 1, 2, 3, 4, ... , n

~~for all the objects~~

For each object we have 2 options (choices) either we will put the object in the bag or we will not put the object in the bag.

1	2	3	4	.....	n
2	2	2	2		2
choices	choices	choices	choices		choices

Total 2^n choices

In our example we 3 objects.

So we have $2^n = 2^3$ choices = 8 choices

Out of these 8 choices 1 ~~each~~ choice will give the optimal solution.

Now how to find out the optimal solution out 8.

~~ob~~

If we go with brute force method.

	ob1	ob2	ob3	Profit
1	0	0	0	No profit
2	0	0	1	60
3	0	1	0	25
4	0	1	1	(85)
5	1	0	0	20
6	1	0	1	80
7	1	1	0	45
8	1	1	1	Not possible as capacity of knapsack is 12.

Example,

weights = { 3, 4, 6, 5 }
Profits = { 2, 3, 1, 4 }
M = 8

Total 4 objects are there soso total we have 2^4 = 16 possibilites

~~Using dynamic programming approach~~

Brute force approach is not always feasible.

Using dynamic programming approach for 0/1 Knapsack problem:-

	i \ w	0	1	2	3	4	5	6	7	8
Pi	Wi	0	0	0	0	0	0	0	0	0
2	3	1	0	0	0	2	2	2	2	2
3	4	2	0	0	0	2	3	3	3	5	5
4	5	3	0	0	0	2	3	4	4	5	6
1	6	4	0	0	0	2	3	4	4	5	6 ←

↓ capacity of knapsack

Wi must be written in ascending order.
so the sequence is {3, 4, 5, 6}

First row & first colm set to zero.

Value of i are the no of objects

For value of (2,4)

~~max(3 + Wi -~~

max((3+ W-Wi), 2)
= max((3+4-4), 2)
= max((3+0), 2)
= max(3,2)
= 3

For value of (2,5)

$= \max((3 + W - W_i), 2)$
$= \max((3 + (5-4)), 2)$

$= \max((3 + (1)), 2)$

check value for 1 in previous step it is 0

$= \max((3+0), 2)$
$= \max(3, 2)$
$= 3$

For value (2,6)

$= \max((3 + W - W_i), 2)$

$= \max((3 + (6-4)), 2)$

$= \max((3 + (2)), 2)$

$= \max((3+0), 2)$
$= \max(3, 2)$
$= 3$

For value of (2,7)

$= \max((3 + (7-4)), 2)$
$= \max((3 + (3)), 2)$
$= \max(3+2, 2)$
$= \max(5, 2)$
$= 5$

For value of (2,8)

$= \max((3 + (8-4)), 2)$
$= \max((3 + (4)), 2)$
$= \max((3+2), 2)$
$= \max(5, 2)$
$= 5$

For value of (3,5)

= max (4 + (5-5), 3)
= max (4 + (0), 3
= max (4, 3)
= 4

For value of (3,6)

= max (4 + (6-5), 3)
= max (4 + (1), 3)
= max (4+0, 3)
= max (4, 3)
= 4

For value of (3,7)

= max ((4 + (7-5)), 5)
= max ((4 + (2)), 5)
= max (4+0, 5)
= max (4, 5)
= 5

For value of (3,8)

= max ((4 + (8-5)), 5)
= max ((4 + (3)), 5)
= max ((4+2), 5)
= max (6, 5)
= 6

For value of (4,6)

= max((1+(6-6)), 4)
= max((1+(0)), 4)
= max((1+0), 4)
= max(1, 4)
= 4

For value of (4,7)

= max((1+(7-6)), 5)
= max((1+(1)), 5)
= max((1+0), 5)
= max(1, 5)
= 5

For value of (4,8)

= max((1+(8-6)), 6)
= max((1+(2)), 6)
= max((1+(0)), 6)
= max((1+0), 6)
= max(1, 6)
= 6

** Formula to calculate value of each cell of matrix m

~~m[i,w] = max(m[i-1,w], m[i-1~~

$$m[i,w] = \max(m[i-1,w], m[i-1, w-w[i]) + P[i])$$

Now how to decide ~~we~~ if we have to select the item or not. Currently pointer is pointing to last col^m of last row.

$x_i = \{ ob1, ob2, ob3, ob4 \}$

check last col^m of last row which is 6 & check last col^m of 2nd last row which is 6

As both of them are same we shift the pointer to 2nd last row & we are not selecting ob4

$x_i = \{ \_, \_, 0, \_ \}$

Now compare

last col^m of 2nd last row	&	last col^m of 3rd last row
6	&	5

As both are not same we select item with ~~profit~~ weight 5 which is present in 2nd last row.

$x_i = \{ \_, \_, 0, 1 \}$

Total Profit = 6

Profit of ob4 ~~is~~ = 4

Remaining Profit = 6 - 4 = 2

For ob1 i.e. 2nd row last colm we have profit of ₹2. So we select ob1

Remaining profit = 2
Profit of ob1 is 2

Now remaining profit = 2 - 2 = 0

$x_i = \{1, 0, 0, 1\}$