CS & IT ENGINEERING

Algorithms

Algorithms

Lecture No. 15

By- Ravindrababu Ravula Sir

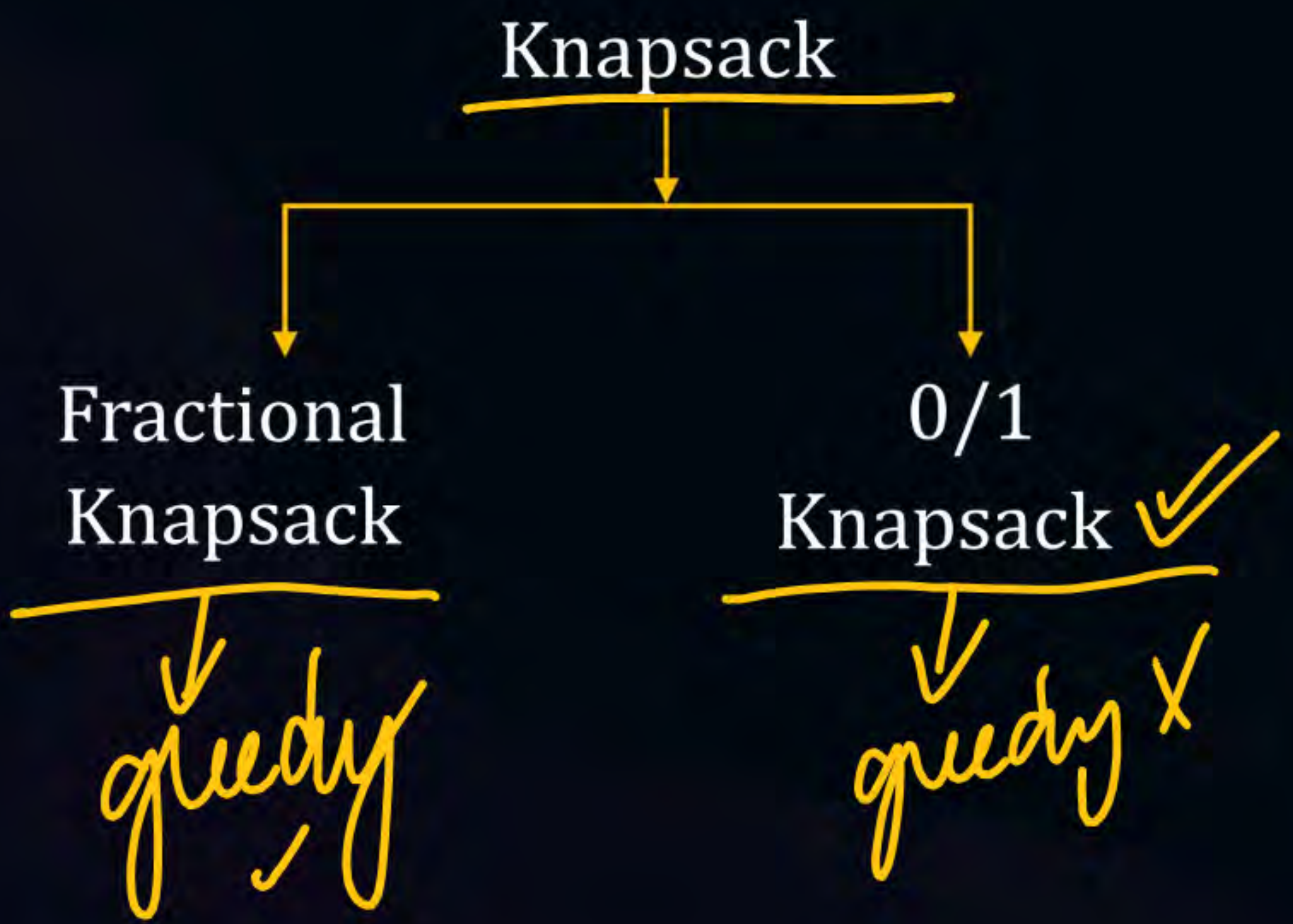
Recap of Previous Lecture

Topic LCS

Topics to be Covered

Topic 01 Knapsack

Topic: Knapsack



Knapsack

Fractional Knapsack

greedy ✓

0/1 Knapsack ✓

greedy ✗

Topic: Knapsack

Difference between fractional and 0/1 knapsack-

- 0/1 Knapsack

Example:

Capacity = 6 units

	✓	✓	✓
Objects →	1	2	3
Weight →	1	2	4
Profit →	10	12	28

Find max profit

Topic: Knapsack

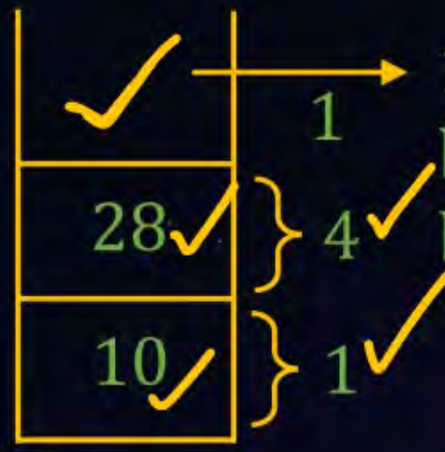
Objects →	1	2	3
Weight →	1	2	4
Profit →	10	12	28

1. Using greedy method:

P/W ratio - 10 6 7 ✓

Put the object according to ~~height~~ P/W ratio.

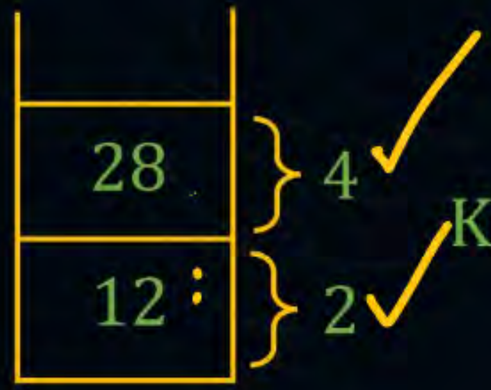
Hence, greedy method fails → (0/1 Knapsack)



1 unit remanning but we can't put 1 unit of object-2 because it is 0/1 Knapsack.

Knapsack

P = 38 ✓



Knapsack full and total profit = 40 ✓

Topic: Knapsack

Example:

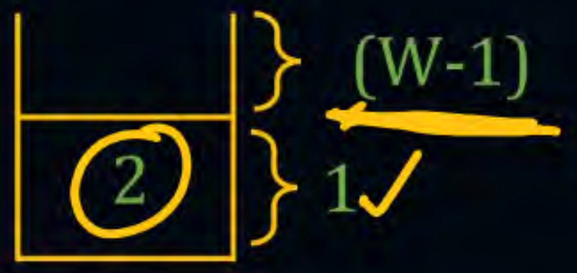
Why greedy method fails in 0 – 1 knapsack

W ✓

W > 1

Objects →	01 ✓	02
~~Weight~~ Profit →	2	~~W~~ 10
~~Profit~~ weight →	1	~~W~~ 10
P/W	2	1

Capacity of Knapsack



Then we can’t put second object.

These why greedy method fails.

Topic: Knapsack

2. Using Dynamic programming to solve 0-1 knapsack problem.

Object: 1 2 3 4 n ✓

$2 \times 2 \times 2 \times 2 \times 2$ Choice put or left alone)

2^n ✓ $O(2^n)$

Topic: Knapsack

2. Using Dynamic programming to solve 0-1 knapsack problem.

Object: 1 2 3 4 n

2 × 2 × 2 × 2 × 2 Choice put or left alone)

2^n

P_i w_i ✓

- Recursive equation –

$$\underset{w}{KS(i,w)} = \begin{cases} \max\left(P_i + KS(i-1, w-w_i), KS(i-1, w)\right); & \boxed{w_i <= w} \ ✓ \\ 0; & i = 0 \text{ or } w = 0 \\ KS(i-1, w); & w_i > w \ ✓ \end{cases}$$

i→ number of the object

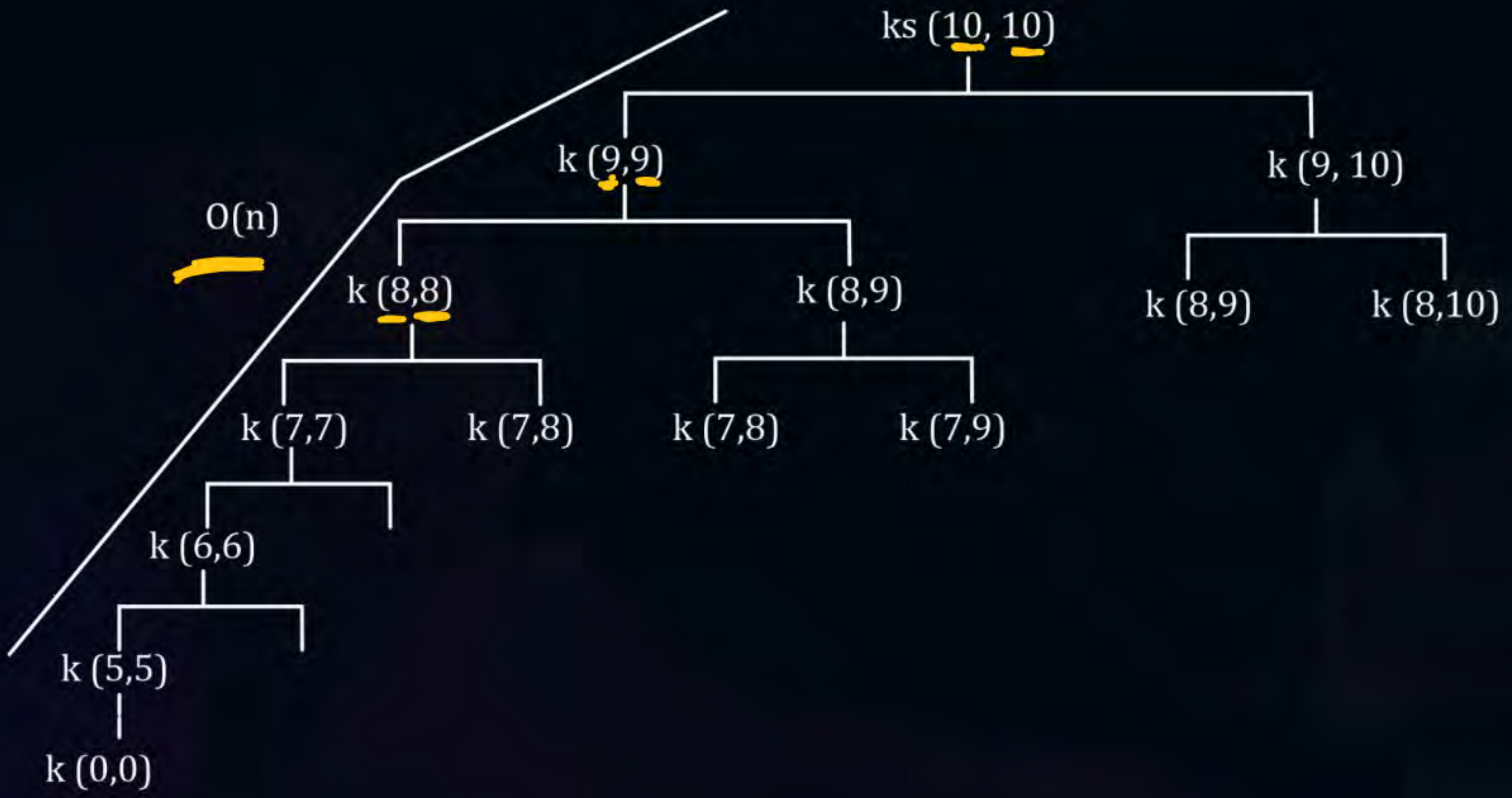
w→ capacity of knapsack

Topic: Knapsack

- Recursion tree for 10 items and Ksize=10 assuming each object wgt is 1

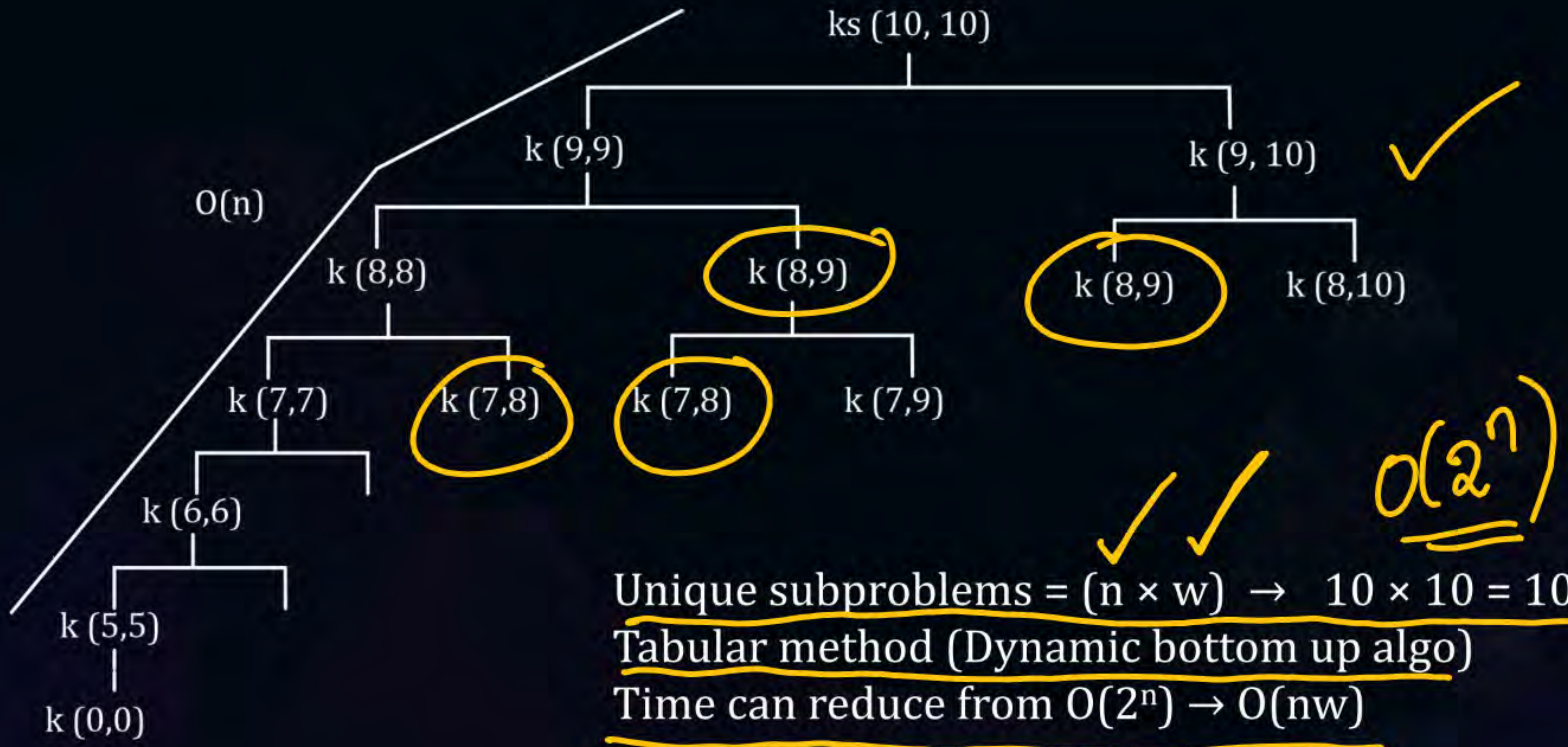
Topic: Knapsack

- Recursion tree for 10 items and Ksize=10 assuming each object wgt is 1



Topic: Knapsack

- Recursion tree for 10 items and Ksize=10 assuming each object wgt is 1



Unique subproblems = (n × w) → 10 × 10 = 100

Tabular method (Dynamic bottom up algo)

Time can reduce from $O(2^n) \rightarrow O(nw)$

Topic: Knapsack

Example:

C = 6

Objects	1	2	3
Weight	1	2	4
Profit	10	12	28

Topic: Knapsack

Example:

C = 6.

Objects →	1	2	3
Weight →	1	2	4
Profit →	10	12	28

$$KS(i,w) = \begin{cases} \max(P_i + KS(i-1, w-w_i), KS(i-1, w)); & w_i <= w \\ 0; & i = 0 \text{ or } w = 0 \\ KS(i-1, w); & w_i > w \end{cases}$$

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

Topic: Knapsack

Example:

C = 6

Objects →	1	2	3
Weight →	1	2	4
Profit →	10	12	28

$$KS(i,w) = \begin{cases} \max\left(P_i + KS(i-1, w-w_i), KS(i-1, w)\right); & w_i <= w \\ 0; & i = 0 \text{ or } w = 0 \\ KS(i-1, w); & w_i > w \end{cases}$$

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

→ Profit max

$$\overset{\checkmark}{KS\,(1, 1)} = \max \begin{cases} \overset{\boxed{10}}{P_1} + \overset{0}{KS\,(0, 0)} \rightarrow \text{Include} \\ \overset{0}{KS\,(0, 1)} \rightarrow \text{Not Include} \end{cases}$$

Topic: Knapsack

Example:

C = 6

Objects →	1	2	3
Weight →	1	2	4
Profit →	10	12	28

$$KS(i,w) = \begin{cases} \max(P_i + KS(i-1, w-w_i), KS(i-1, w)); & w_i <= w \\ 0; & i = 0 \text{ or } w = 0 \\ KS(i-1, w); & w_i > w \end{cases}$$

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

→ Profit max

$$KS(\overset{\checkmark}{\underline{1}}, \underline{2}) = \max \begin{cases} \overset{\checkmark}{\overset{10}{\underline{P_1}}} + \underline{KS(\overset{0}{0, 1})} \to 10 \\ \underset{0}{KS(0, 2)} \end{cases}$$

Topic: Knapsack

Example:

C = 6

Objects →	1	2	3
Weight →	1	2	4
Profit →	10	12	28

$$KS(i,w) = \begin{cases} \max(P_i + KS(i-1, w-w_i), KS(i-1, w)); & w_i <= w \\ 0; & i = 0 \text{ or } w = 0 \\ KS(i-1, w); & w_i > w \end{cases}$$

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

→ Profit max

$$KS(\underline{1}, \underline{3}) = \max \begin{cases} \overset{10}{P_1} + \overset{0}{KS(\underline{0}, 2)} \\ \overset{0}{KS(\underline{0}, 3)} \end{cases}$$

Topic: Knapsack

Example:

$C = 6$

Objects →	1	2	3
Weight →	1	2	4
Profit →	10	12	28

$$KS(i,w) = \begin{cases} \max(P_i + KS(i-1, w-w_i), KS(i-1, w)); & w_i <= w \\ 0; & i = 0 \text{ or } w = 0 \\ KS(i-1, w); & w_i > w \end{cases}$$

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

🟨 → Profit max

$$\underline{KS(1, \underline{4})} = \max \begin{cases} \overset{10 \checkmark}{\underline{P_1}} + \underline{\overset{0}{KS(0, 3)}} \\ \overset{0}{KS(0, 4)} \end{cases}$$

Topic: Knapsack

Example:

C = 6

Objects →	1	2	3
Weight →	1	2	4
Profit →	10	12	28

$$KS(i,w) = \begin{cases} \max\left(P_i + KS(i-1, w-w_i), KS(i-1, w)\right); & w_i <= w \\ 0; & i = 0 \text{ or } w = 0 \\ KS(i-1, w); & w_i > w \end{cases}$$

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

→ Profit max

$$KS(1, 5) = \max \begin{cases} \overset{10}{P_1} + \overset{0}{KS}(0, 4) \\ \overset{0}{KS}(0, 5) \end{cases}$$

Topic: Knapsack

Example:

C = 6

Objects →	1	2	3
Weight →	1	2	4
Profit →	10	12	28

$$KS(i,w) = \begin{cases} \max\left(P_i + KS(i-1, w-w_i), KS(i-1, w)\right); & w_i <= w \\ 0; \; i = 0 \text{ or } w = 0 & \\ KS(i-1, w); & w_i > w \end{cases}$$

	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						

→ Profit max

$$KS(1, 6) = \max \begin{cases} \overset{10}{P_1} + \overset{0}{KS(0, 5)} \\ \overset{0}{KS(0, 6)} \end{cases}$$

Topic: Knapsack

Example:

C = 6

Objects →	1	2	3
Weight →	1	2	4
Profit →	10	12	28

$$KS(i,w) = \begin{cases} \max(P_i + KS(i-1, w-w_i), KS(i-1, w)); & w_i <= w \\ 0; & i = 0 \text{ or } w = 0 \\ KS(i-1, w); & w_i > w \end{cases}$$

	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					
3	0						▨

→ Profit max

$$K(2, 1) \checkmark = \begin{cases} \cancel{M_i > m \quad 2>1} \\ K(1, 1) \end{cases}$$

Topic: Knapsack

Example:

C = 6

Objects →	1	2	3
Weight →	1	2	4
Profit →	10	12	28

$$KS(i,w) = \begin{cases} \max(P_i + KS(i-1, w-w_i), KS(i-1, w)); & w_i <= w \\ 0; & i = 0 \text{ or } w = 0 \\ KS(i-1, w); & w_i > w \end{cases}$$

	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	12				
3	0						

→ Profit max

$$\overset{\checkmark}{K(2,2)} = \begin{cases} \overset{12 \checkmark}{P_2} + \overset{0 \checkmark}{KS(1,0)} \\ \overset{10}{KS(1,2)} \end{cases}$$

Topic: Knapsack

Example:

$C = 6$

Objects →	1	2	3
Weight →	1	$\underline{2}$	4
Profit →	10	12	28

$$KS(i,w) = \begin{cases} \max(P_i + KS(i-1, w-w_i), KS(i-1, w)); & w_i <= w \\ 0; & i = 0 \text{ or } w = 0 \\ KS(i-1, w); & w_i > w \end{cases}$$

	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	12	$\underline{22}$			
3	0						▨

→ Profit max

$$K(2, \underline{3}) = \begin{cases} \overset{12}{P_2} + \overset{10}{KS(\underline{1},\underline{1})} ✓ \\ \overset{10}{KS(\underline{1}, \underline{3})} \end{cases}$$

Topic: Knapsack

Example:

C = 6

Objects →	1	2	3
Weight →	1	2	4
Profit →	10	12	28

$$KS(i,w) = \begin{cases} \max\left(P_i + KS(i-1, w-w_i), KS(i-1, w)\right); & w_i <= w \\ 0; & i = 0 \text{ or } w = 0 \\ KS(i-1, w); & w_i > w \end{cases}$$

	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	12	22	22		
3	0						■

→ Profit max

$$K(2,4)\checkmark = \begin{cases} \overset{\boxed{12}}{\underline{P_2}} + \overset{10\ \checkmark}{\underline{KS(1,2)}}\ \checkmark \\ \overset{10}{\underline{KS(1,4)}} \end{cases}$$

Topic: Knapsack

Example:

C = 6

Objects →	1	2	3
Weight →	1	2	4
Profit →	10	12	28

$$KS(i,w) = \begin{cases} \max(P_i + KS(i-1, w-w_i), KS(i-1, w)); & w_i <= w \\ 0; & i = 0 \text{ or } w = 0 \\ KS(i-1, w); & w_i > w \end{cases}$$

	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	12	22	22	22	
3	0						▨

→ Profit max

$$K(\underline{2}, \overset{\checkmark}{\underline{5}}) = \begin{cases} \underline{\overset{\boxed{12}}{P_2}} + \overset{\boxed{10}}{KS(\underline{1},\underline{3})} \\ \overset{\boxed{10}\ \checkmark}{KS(\underline{1}, \underline{5})} \end{cases}$$

Topic: Knapsack

Example:

C = 6

Objects →	1	2	3
Weight →	1	2	4
Profit →	10	12	28

$$KS(i,w) = \begin{cases} \max\left(P_i + KS(i-1, w-w_i), KS(i-1, w)\right); & w_i <= w \\ 0; & i = 0 \text{ or } w = 0 \\ KS(i-1, w); & w_i > w \end{cases}$$

	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	12	22	22	22	22
3	0						■

→ Profit max

$$K(\underline{2}, \underline{6}) \overset{\checkmark}{=} \begin{cases} \underline{\overset{12}{P_2}} + \underline{KS(1,\overset{10}{4})} \\ KS\,(\underline{1}, \overset{10}{\underline{6}}) \end{cases}$$

Topic: Knapsack

Example:

C = 6

Objects →	1	2	3
Weight →	1	2	4 ✓
Profit →	10	12	28

$$KS(i,w) = \begin{cases} \max(P_i + KS(i-1, w-w_i), KS(i-1, w)); & w_i <= w \\ 0; & i = 0 \text{ or } w = 0 \\ KS(i-1, w); & w_i > w \end{cases}$$

	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	12	22	22	22	22
✓ 3	0	10					■

■ → Profit max

$$K(\underline{\overset{\times}{3}}, \underline{\overset{\checkmark}{1}}) = \begin{cases} \\ KS(\underline{2}, \underline{\overset{\boxed{10}}{1}}) \end{cases}$$

Topic: Knapsack

Example:

$C = 6$

Objects →	1	2	3
Weight →	1	2	$\underline{4}$
Profit →	10	12	28

$$KS(i,w) = \begin{cases} \max(P_i + KS(i-1, w-w_i), KS(i-1, w)); & w_i <= w \\ 0; \; i = 0 \text{ or } w = 0 \\ KS(i-1, w); \; w_i > w \end{cases}$$

	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	12	22	22	22	22
✓3	0	10	12 ✓				▨

▨ → Profit max

$$K(\underline{3}, \underline{2}) = \begin{cases} \\ \overset{\boxed{12}}{KS(\underline{2}, \underline{2})} \end{cases}$$

Topic: Knapsack

Example:

C = 6

Objects →	1	2	3
Weight →	1	2	4
Profit →	10	12	28

$$KS(i,w) = \begin{cases} \max (P_i + KS(i-1, w-w_i), KS(i-1, w)); & w_i <= w \\ 0; \; i = 0 \text{ or } w = 0 & \\ KS(i-1, w); \; w_i > w & \end{cases}$$

	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	12	22	22	22	22
✓3	0	10	12	22			

→ Profit max

$$\overset{\checkmark}{K(\underline{3}, \underline{3})} = \begin{cases} \\ \overset{22}{KS(\underline{2}, \underline{3})} \end{cases}$$

Topic: Knapsack

Example:

C = 6

Objects	1	2	3
Weight	1	2	4
Profit	10	12	28

$$KS(i,w) = \begin{cases} \max(P_i + KS(i-1, w-w_i), KS(i-1, w)); & w_i <= w \\ 0; & i = 0 \text{ or } w = 0 \\ KS(i-1, w); & w_i > w \end{cases}$$

	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	12	22	22	22	22
✓ 3	0	10	12	22	✓ 28		▨

▨ → Profit max

$$\overset{\checkmark\ \checkmark}{K(3, 4)} = \begin{cases} \overset{28}{P_3} + \overset{0}{KS(2,0)} \ \checkmark \\ \overset{22}{KS(2, 4)} \end{cases}$$

Topic: Knapsack

Example:

C = 6

Objects →	1	2	3
Weight →	1	2	4
Profit →	10	12	28

$$KS(i,w) = \begin{cases} \max(P_i + KS(i-1, w-w_i), KS(i-1, w)); & w_i <= w \\ 0; & i = 0 \text{ or } w = 0 \\ KS(i-1, w); & w_i > w \end{cases}$$

	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	12	22	22	22	22
✓ 3	0	10	12	22	28	✓ 38	

→ Profit max

$$K(\overset{\checkmark}{3}, \overset{\checkmark}{5}) = \begin{cases} \underbrace{\overset{28}{P_3} + \overset{10}{KS(2,1)}}_{22} \\ KS(2, 5) \end{cases}$$

Topic: Knapsack

Example:

C = 6

Objects →	1	2	3
Weight →	1	2	4
Profit →	10	12	28

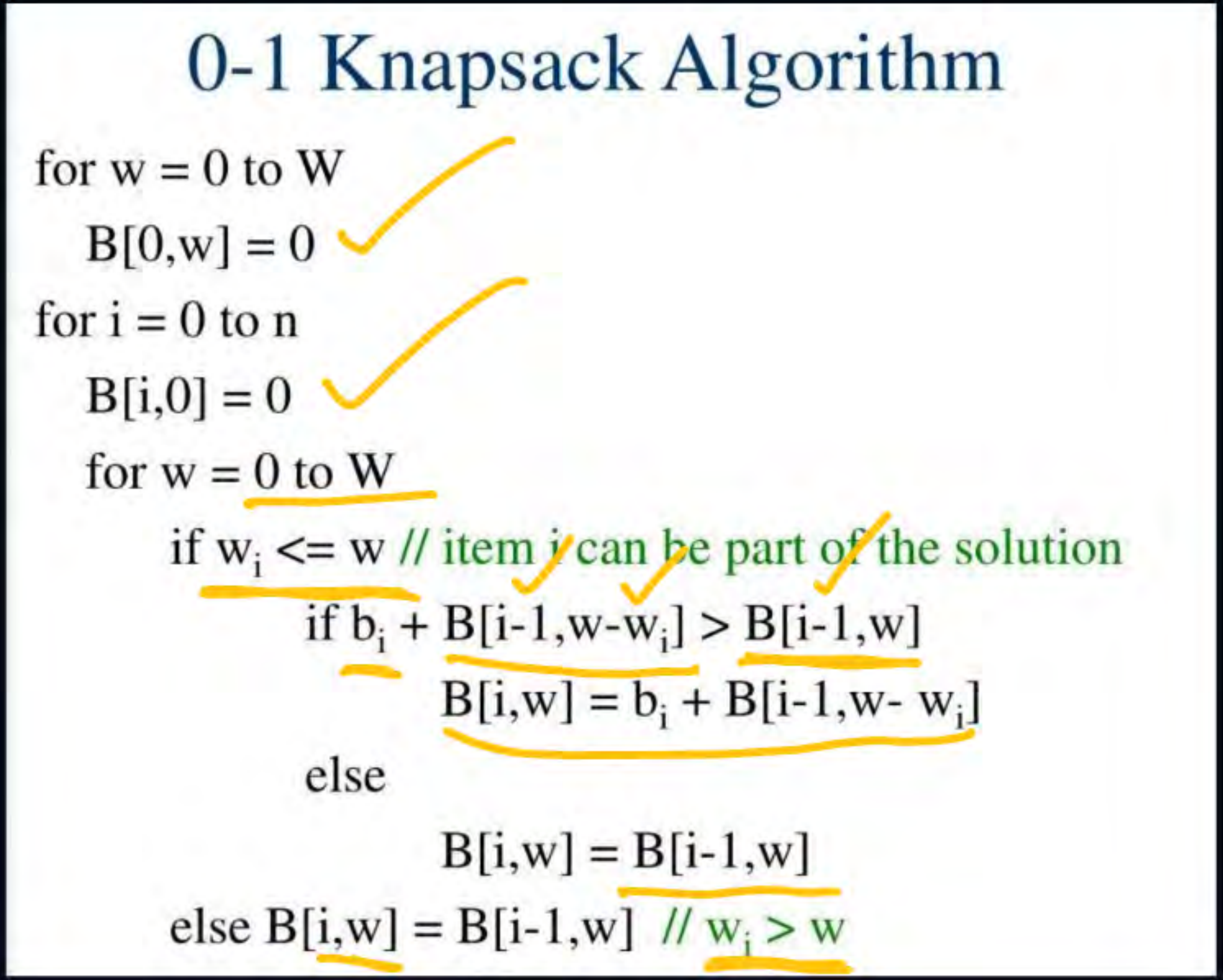
$$KS(i,w) = \begin{cases} \max(P_i + KS(i-1, w-w_i), KS(i-1, w)); & w_i <= w \\ 0; \ i = 0 \text{ or } w = 0 & \\ KS(i-1, w); & w_i > w \end{cases}$$

	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	12	22	22	22	22
✓ 3	0	10	12	22	28	38	**40** ✓

→ Profit max

$$K(\underline{3}, \overset{-4}{\underline{6}}) = \begin{cases} \overset{28}{P_3} + \underline{KS(\overset{12}{2,2})} \\ KS(\overset{22}{\underline{2}, \underline{6}}) \end{cases}$$

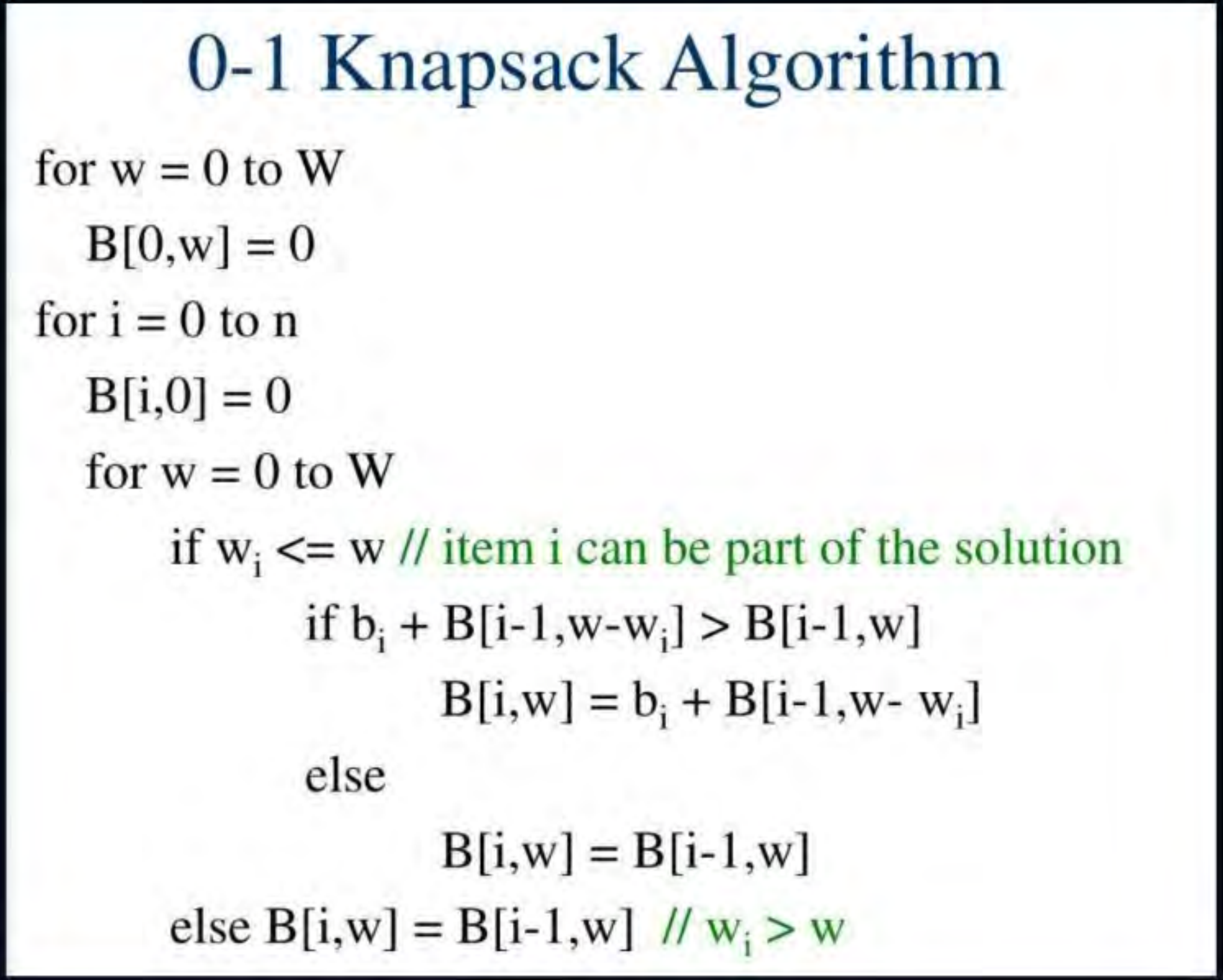
Topic: Knapsack



0-1 Knapsack Algorithm

```
for w = 0 to W
   B[0,w] = 0
for i = 0 to n
   B[i,0] = 0
   for w = 0 to W
        if $w_i$ <= w // item i can be part of the solution
                if $b_i$ + B[i-1,w-$w_i$] > B[i-1,w]
                        B[i,w] = $b_i$ + B[i-1,w- $w_i$]
                else
                        B[i,w] = B[i-1,w]
        else B[i,w] = B[i-1,w]  // $w_i$ > w
```

Topic: Knapsack



0-1 Knapsack Algorithm

```
for w = 0 to W
   B[0,w] = 0
for i = 0 to n
   B[i,0] = 0
   for w = 0 to W
        if $w_i$ <= w // item i can be part of the solution
                if $b_i$ + B[i-1,w-$w_i$] > B[i-1,w]
                        B[i,w] = $b_i$ + B[i-1,w- $w_i$]
                else
                        B[i,w] = B[i-1,w]
        else B[i,w] = B[i-1,w]  // $w_i$ > w
```

Time complexity = O(nw) ✓✓

Space complexity = O(nw)

T(n) = min(O(2^n), O(nw)) ✓

$$W = O(2^n)$$

THANK - YOU