DP 4 : Applications Of Knapsack

↳ Category

AGENDA

- Rod Cutting
- Coin Change
- 0-1 Knapsack-2

Quiz :- Time complexity of the Unbounded 0-1 knapsack problem ?

→ W : capacity of knapsack
→ N : No. of elements
→ K : max weight of any item
→ P : max value of any item

↳ TC :- O(N * W)

QUICK RECAP

↳ items { → weight, → value

cap □ [maximize value/profit]

→ 01 KNAPSACK

↳ can pick one element one time

→ Unbounded Knapsack

↳ Element can be picked multiple times.

Q How to figure out knapsack problem ?

↳ Choose among items → objects

↓

array Elements

↳ Constraints [Mandatory]

↳ Maximize or minimize

Question :- **Rod Cutting problem**

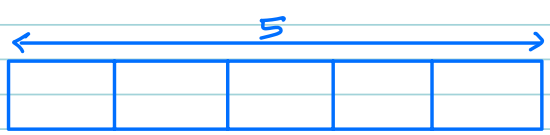
↳ A rod of length N & an array A of length N is given. The elements (i) of the array represents value the can be made by selling (i+1) length of the rod. Find the maximum values that can be obtained by cutting the rod into some peices & selling them.

Ex1:- N = 5

Cut of this length

A = [0, 1, 4, 2, 5, 6]
0 1 2 3 4 5

Get profit as this



Ex :-

① 2 + 3
(4) (2) → 6

② 4 + 1
(5) (1) → 6

③ 2 + 2 + 1
(4) (4) (1) → 9

④ 3 + 1 + 1
(2) (1) (1) → 4

⑤ 5
(6) → 6

⑥ 1 + 1 + 1 + 1 + 1
(1) (1) (1) (1) (1) → 5

Quiz 2 :- The Cutting rod is :-

↳ Unbounded Knapsack

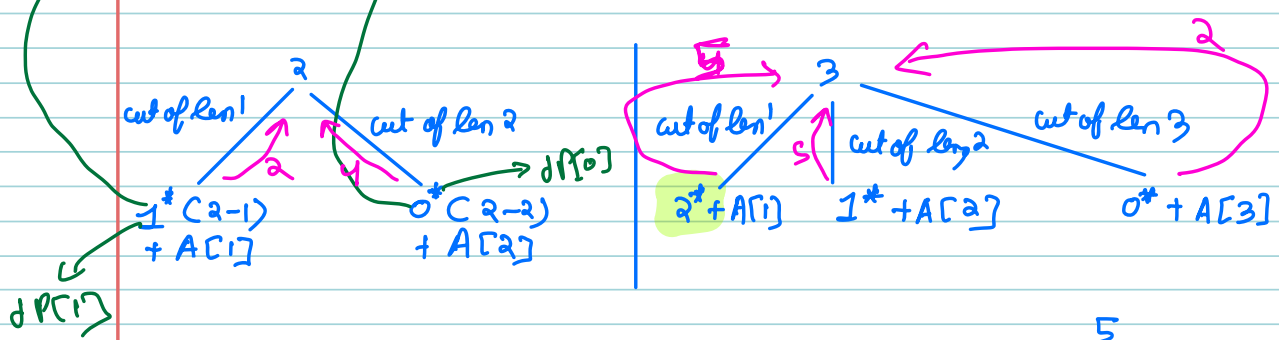
Q what we will be storing in dP[i] ?

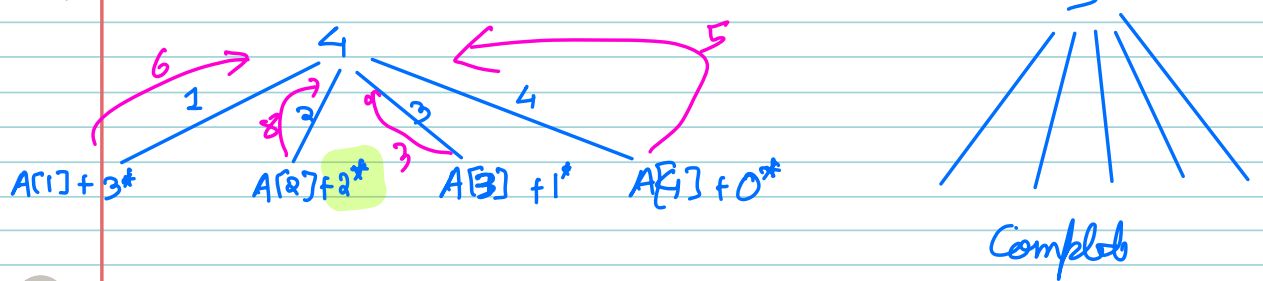
↳ dP[i] = Max profit I can get with rod of length "i".

Ex :- N = 5

A = [0, 1, 4, 2, 5, 6]
(0 1 2 3 4 5)

	0	1	2	3	4	5
dP →	0	1	4	5	8	9
		1	2	1,2	2,2	1,2,2





Memoization ⟹ H.W.

TABULATION

```
dp[N+1] = {0};
dp[0] = 0
for (i = 1 to N){          → Rod
    for (j = 1 to i){      → cuts
        dp[i] = max(dp[i], A[j]+dp[i-j]);
    }
}
return dp[N];
```

TC — $O(N^2)$

SC — $O(N)$

Question 2 :- COIN CHANGE PERMUTATION

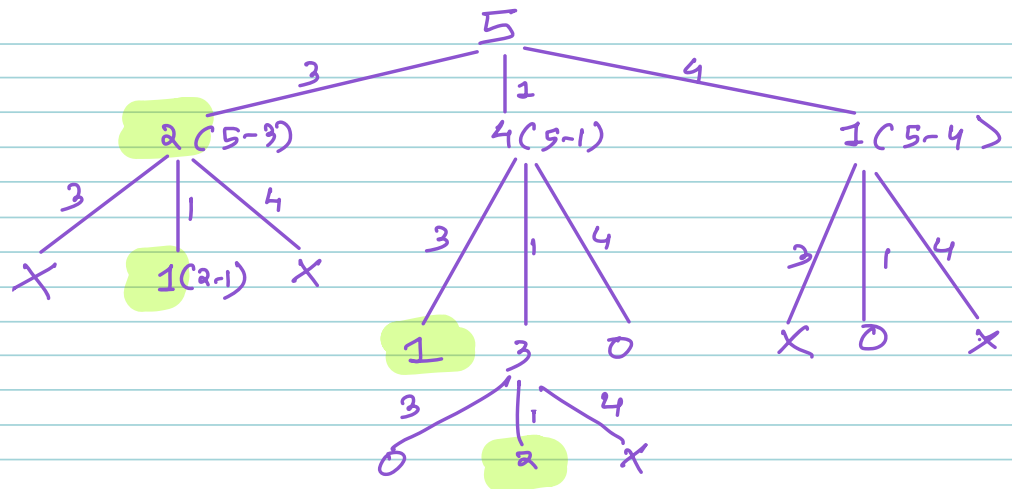
In how many ways can the sum be equal to N by using coins given in the array ? One coin can be used multiple times.

Note :- $(x, y) \neq (y, x)$

Ex :- A = {3, 1, 4} ← Denomination present

K = 5

Ans :- 6 Way [{1, 4} {4, 1}, {1 1 1 1 3 {3, 1, 1} {1, 3, 1} {1, 1, 3}]



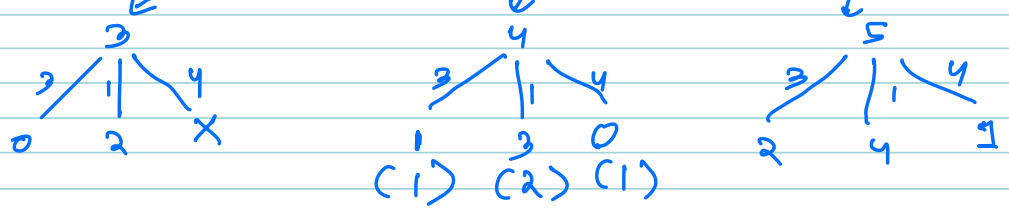
Can we use DP ?
↳ yes

what will be the dimention of dp array ?
↳ 1D

what we will be storing in dp[i] ?
↳ dp[i] = No. of Permutations (ways) to pay i rupees.

Quiz :- what is the no. of ways to get sum = 0
↳ 1

0	1	2	3	4	5
1	1	1	2	4	6
	(1)	(1 1)	(3 / 1 1 1)	1 3 3 1 1 1 1 1 4	1 1 3 1 3 1 3 1 1 1 1 1 1 1 4 1 1 4



PSEUDO CODE

```
dp[k+1] = {0};
dp[0] = 1;
for ( i -> 1 to k){          // k -> Sum
    for (j -> 1 to N){       // N -> coins array
        if ( A[j] <= i){
            dp[i] += dp[ i - A[j]];
        }
    }
}
```

TC - O(K * N)

SC - O(K)

Question 3:- Coin Change Combination

Given a set of coins & a target sum. Find the no. of ways to make the target sum using the coins, where each order can only be used once.

$(x, y) = (y, x)$

Eg K = 5 , A = [3, 1, 4]

Ans = 3 [(1, 4) (1, 1, 1, 1, 1), (3, 1, 1)]

Q what dp[i] will be storing

↳ dp[i] = No. of Combinations to pay "i" sum

Ex:- A = [3, 1, 4] | k = 5

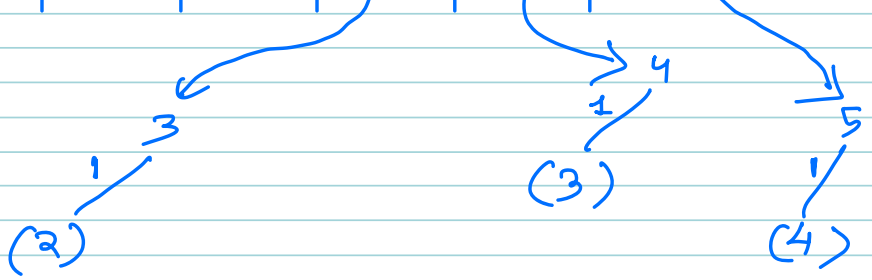
IDEA :-

Step1:- We will go over the array & see how many ways are there to form 'i'th amount with denomination "3"

0	1	2	3	4	5
1	0	0	1	0	0
			(3)		

Step2:- We will go over the array & see how many ways are there to form 'i'th amount with denomination "1"

0	1	2	3	4	5
1	1 ∅	1 ∅	∅ 2	∅ 2	∅ 2
	(1)	(11)	(3) (111)	(31 1111)	(311 1111)



Q Why we require ~~to denominations~~ two steps 1 by 1 over here ?

↳ So that we dont Stuck in Scenario of 13 & 31. With this approach it will not allow 3 to come later.

Steps:- We will go over the array & see how many ways are there to form ith amount with denomination 4

0	1	2	3	4	5
1	1 ∅	1 ∅	~~1~~ 2	3 ~~∅~~ ~~2~~	∅ ~~2~~ ~~3~~
	(1)	(11)	(3) (111)	(31 1111)	(311 1111 (14)

4 → 4 → 4 → 0(4-4)

5 → 4 → 1(5-4)

PSEUDO CODE

```
dp[k+1] = {0};
dp[0] = 1;
for (j -> 1 to N)        // coins array
    for (i -> 1 to k){   // sum
        if (A[j] <= i){
            dp[i] += dp[i - A[j]];
        }
    }
}
```

TC - O(N*K)

SC - O(K)

Question 4 :- 0-1 KNAPSACK FOLLOW-UP

We are given N toys with their happiness & weight. Find total happiness that can be kept in a bag with the capacity W. Here we cannot divide toys.

CONSTRAINTS

$1 <= N <= 500$

$1 <= h[i] <= 50$

$1 <= wt[i] <= 10^9$

$1 <= W <= 10^9$

Quiz :- what is the max value we can get for these items i.e. (weight, value) pairs in 0-1 Knapsack of

Capacity = 8

Items = [(3, 12), (6, 20), (5, 15), (2, 6), (4, 10)] (weight → first, value → second)

↳ 27

⇒ Approach learnt in last class

TC → O(N * W)

500 * 10^9

⇒ 5×10^{11}

Not possible

dp[i][j] = Max profit we can get in a bag of capacity "j", if we choose from first i items.

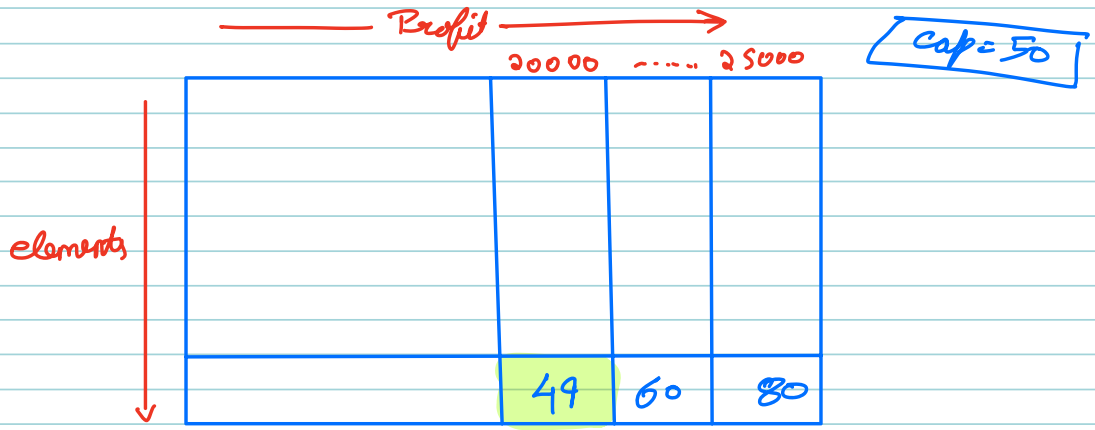
Now Total Profit Based Constraints

↓

500 × 50 = 2500

So Let Update what is Stored in dp[i][j] & Explore.

↳ dp[i][j] = min weight required to get Val J if only first "i" items are required.



Ans = The last first row profit for from Right Side in which dp[i][j] <= Cap.