Agenda

1. Cutting the Rod
2. Coin Change I / II
3. Extended 0-1 Knapsack

① Fractional → Greedy

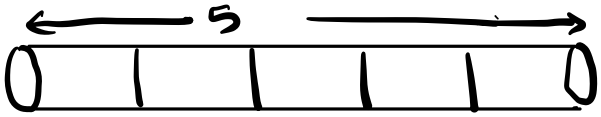
② 0-1 → DP

③ $0-\infty$ / unbounded → DP

1. Given a rod of length N, and an array of len N, where A[i] = price of (i+1) len rod. Find maximum val we can obtain by selling the rod.

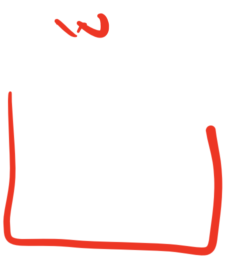
idx	0	1	2	3	4
	1	2	3	4	5

N = 5 A = [1, 4, 2, 5, 6]



ans → 9

length	Total Value
5	6
4+1	5+1=6
3+2	2+4 =6
3+1+1	2+1+1 = 4
2+2+1	4+4+1 =9
1+1+1+1+1	1+1+1+1+1 =5
2+1+1+1	4+1+1+1=7



W → len of rod

wt[i] → len of individual rod

Max → val of rod

$\downarrow$

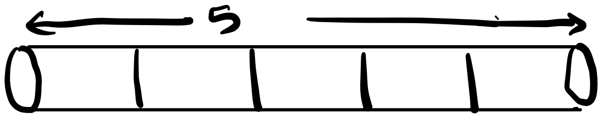
A[i]

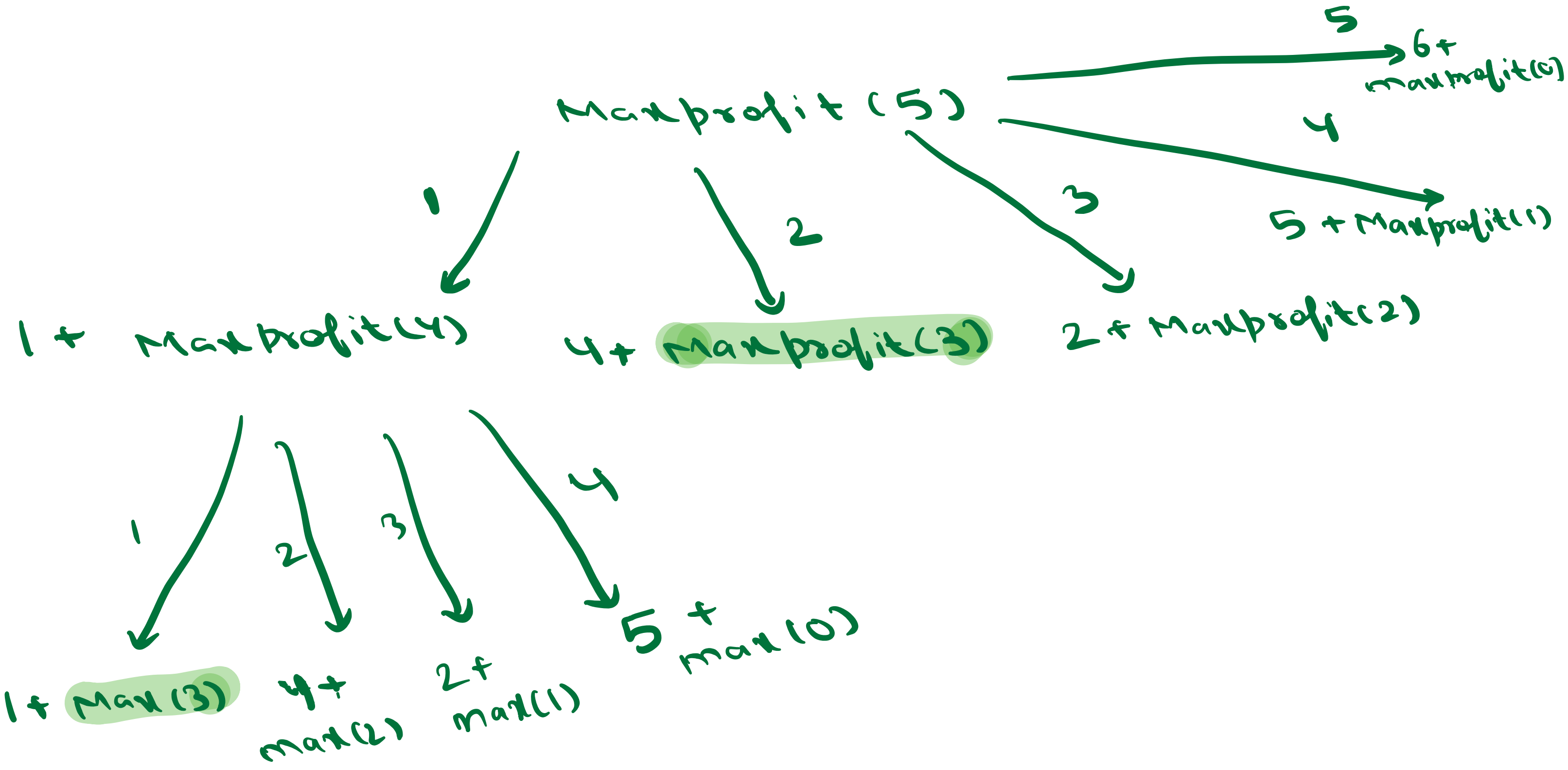
0-1 / Unbounded Knapsack

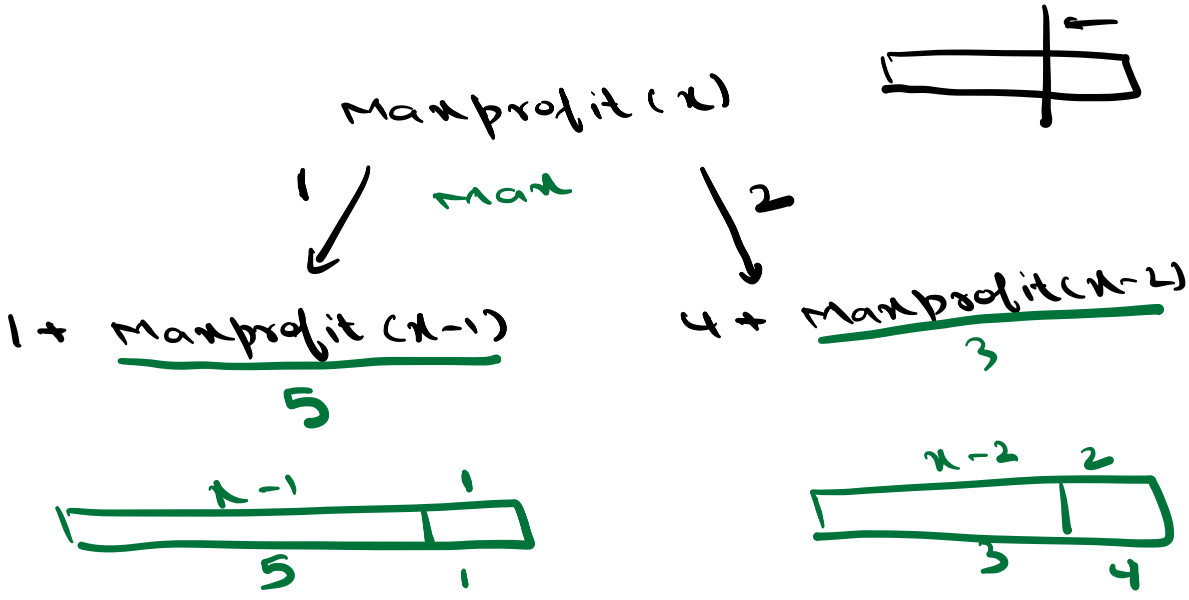
idx	0	1	2	3	4
	1	2	3	4	5

N = 5

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







int dp[N+1] <0, 1, 2 . . . . N>

dp[i] → max. profit we can get by selling a rod of len i

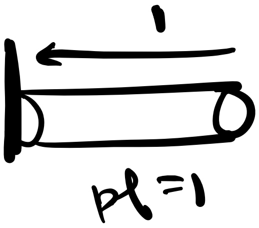
N = 5

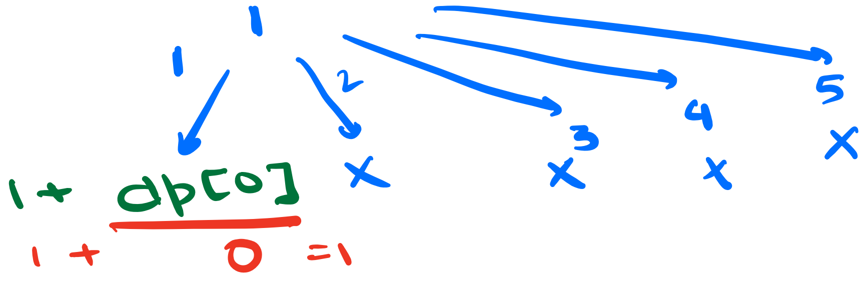
dp[5+1] = 6

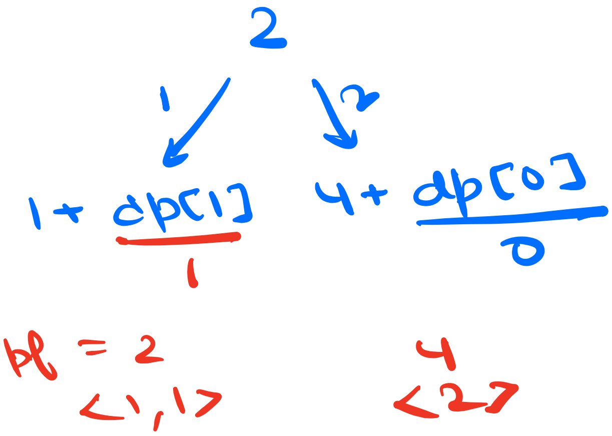
idx	0	1	2	3	4
	1	2	3	4	5

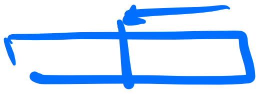
A = [1, 4, 2, 5, 6]

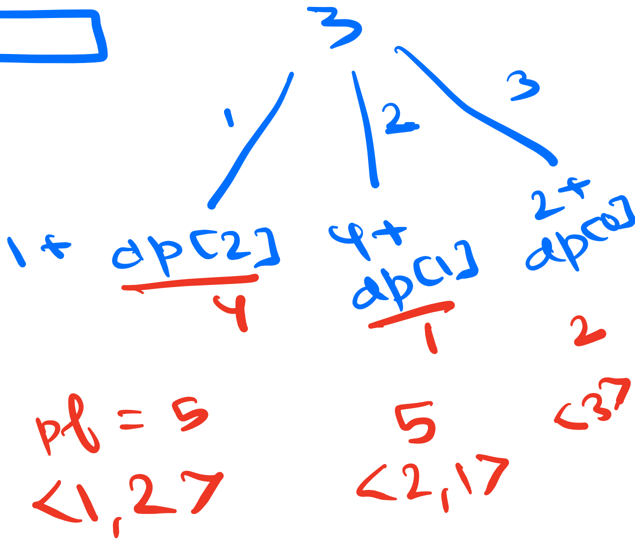
idx	0	1	2	3	4	5
dp[]	0	1 <1>	4 <2>	5 <2,1>	8 <2,2>	9 <1, 2, 2> <2, 2, 1>







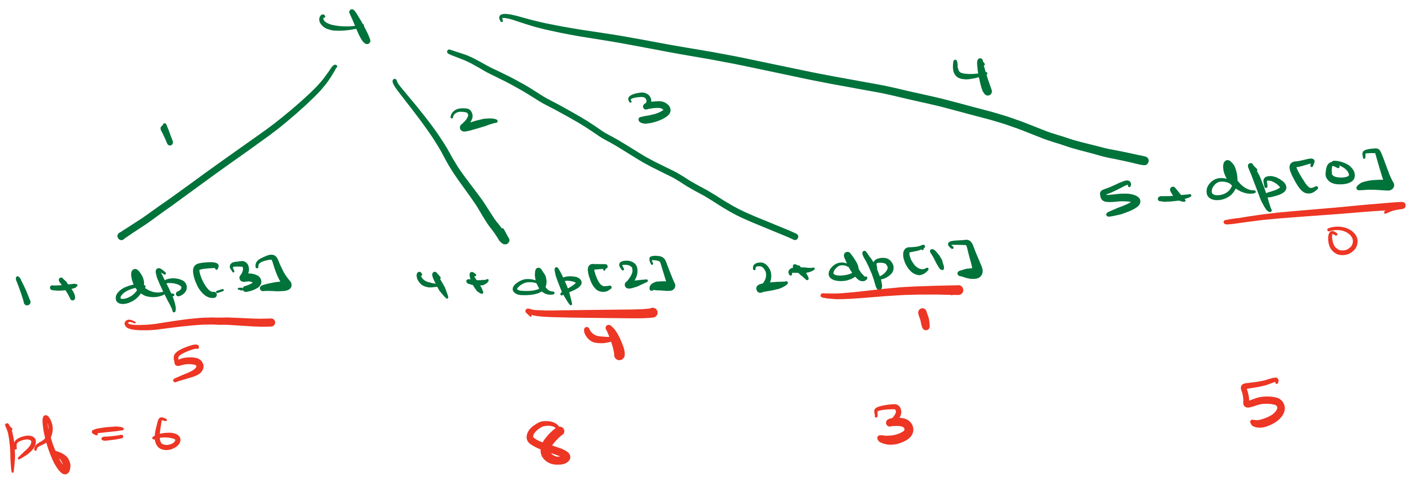


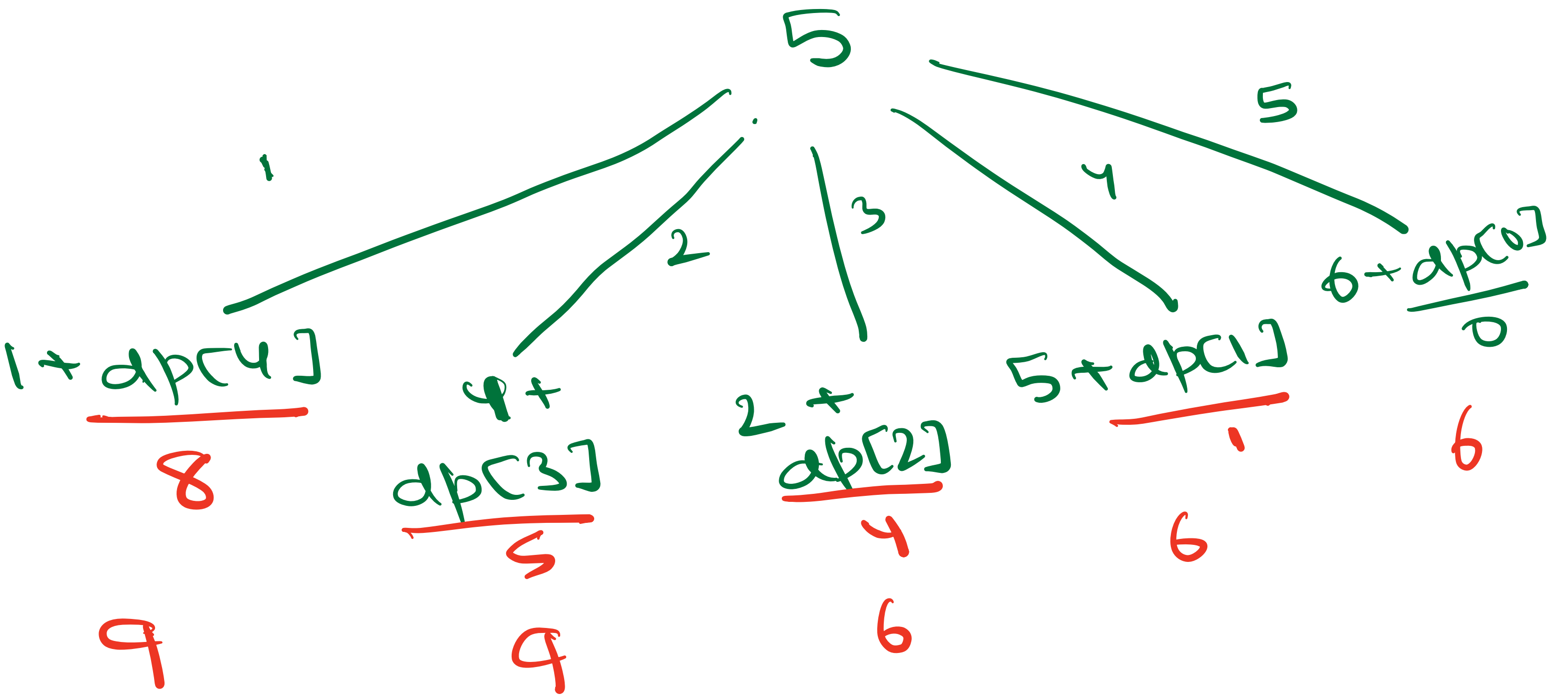


idx 0 1 2 3 4

1 2 3 4 5

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





N=5 < 1, 3, 5 >

```
int dp[N+1] = <0>

for (len = 1 ; len ≤ N ; len ++) <
    int maxval = -∞/0
    for ( cut = 1 ; cut ≤ len ; cut++) <
        maxval = max (maxval, A [cut -1] + dp[len-cut])
    >
    dp[len] = maxval
>

return dp[N]
```

TC : $O(N^2)$

SC : $O(N)$

2. Coin Change (Permutation)

$(x,y) \neq (y,x)$

↓

Order of coins does matter

K = 5   A = [3, 1, 4]   ans = 6

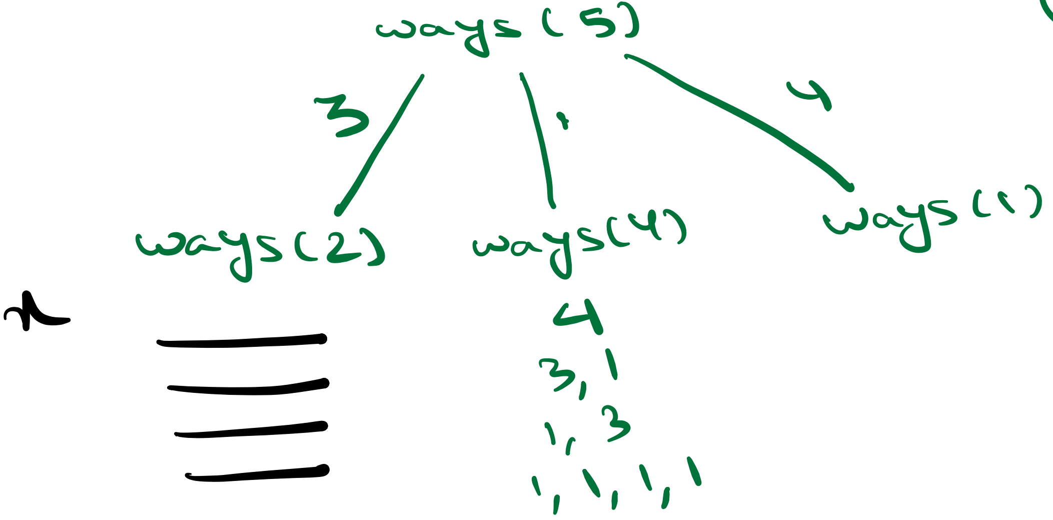
In how many diff ways can we make Rs k?

<1, 4>   <4, 1>

<3, 1, 1>   <1, 3, 1>   <1, 1, 3>

<1, 1, 1, 1, 1>

3, 1, 4



ways(5) = ways(2) + ways(4) + ways(1)

int dp[k+1] = <0>

dp[0] = 1

K = 5

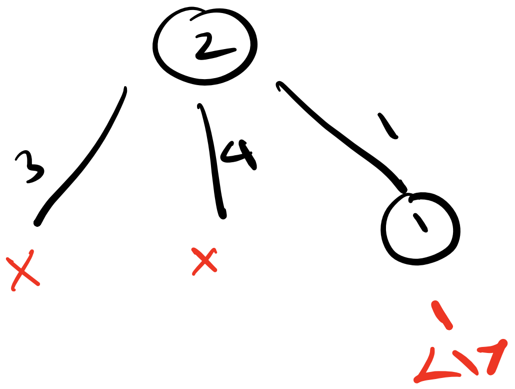
A = [3, 1, 4]

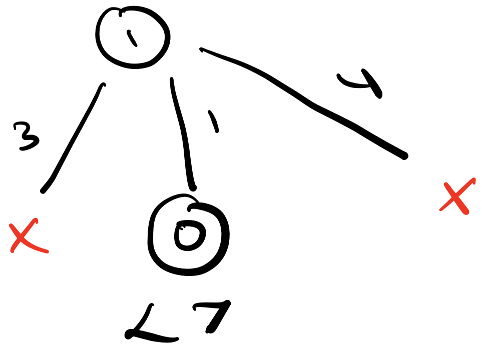
int dp[6] = <0>

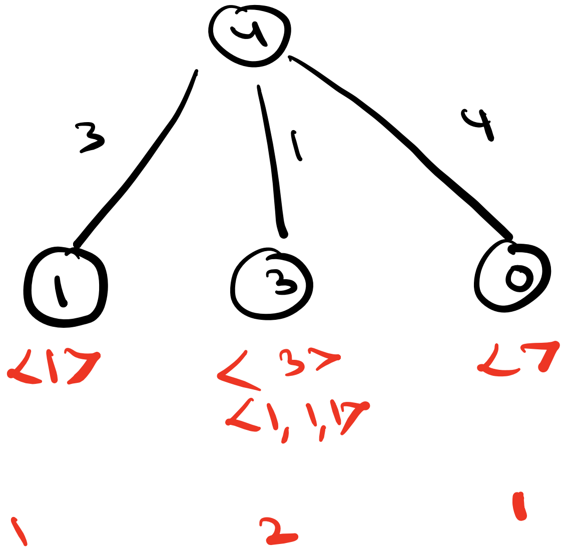
dp[0] = 1

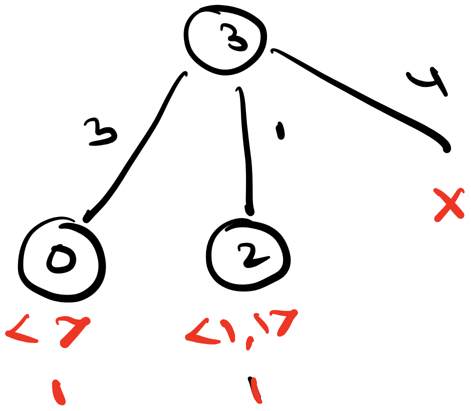
dp[i] = no. of ways to make Rs. i

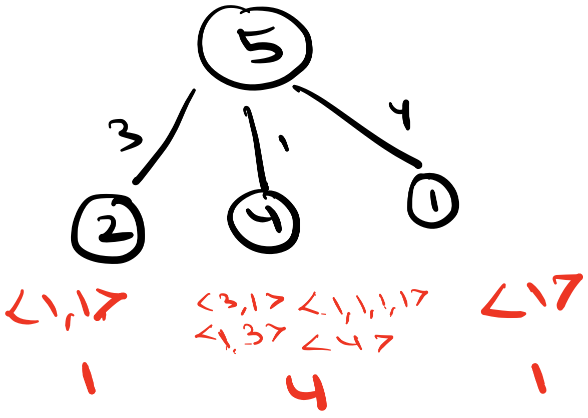
idx	0	1	2	3	4	5
dp	1	1	1	2	4	6
		<1>	<1,1>	<3> <1,1,1>	<3,1> <1,3> <1,1,1,1> <4>	







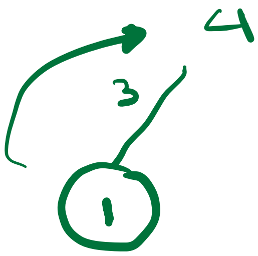




```
int dp[K+1] = <0>
    dp[0] = 1     (don't select anything)

for ( amt = 1 ; amt ≤ K ; amt ++ ) <

    for ( j=0 ; j < coin.len ; j++ ) <
        if ( coin[j] ≤ amt )
            dp[amt] += dp[amt - coin[j]]
    >
>
return dp[K]
```



TC : O (N * K)

no of coins

₹

SC : O(K)

10 : 37

3. Coin Change (Combination)

(x,y) = (y,x)
↓
Order of coins does not matter

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

<1,4>   <4,1>

<3, 1, 1>   <1, 3, 1>   <1,1,3>

<1,1,1,1,1>

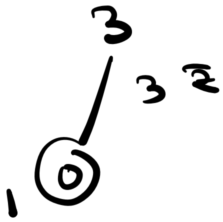
ways (5)

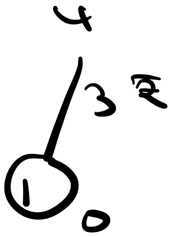
dp [i] = no. of combinations to make i Rs.

0 1 2

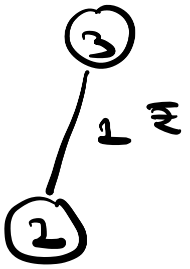
K = 5   A = [3, 1, 4]

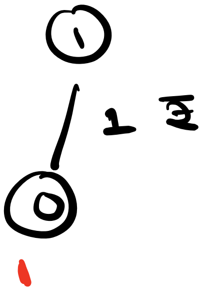
dp [6]

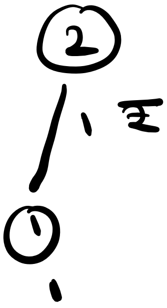


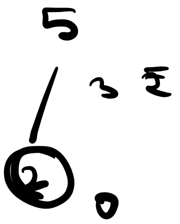


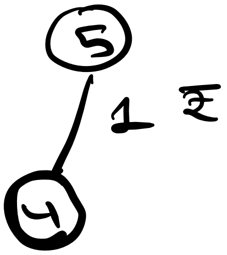
idx	0	1	2	3	4	5
dp	1	~~0~~ 1	~~0~~ 1	~~0~~ ~~1~~ 2	~~0~~ ~~2~~ 3	~~0~~ ~~2~~ 3
		<1>	<1, 1>	<3> <1,1,1>	<3,1> <1,1,1,1> <4>	<3,1,1> <1,1,1,1,1> <1,4>



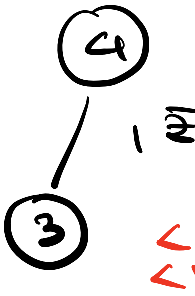








2



<3>
<1,1,1>

```
      int  dp[K+1]  = {0}
         dp[0] = 1      (don't select anything)

for ( j=0 ; j < coin.len ; j++) {

    for (amt =1 ; amt ≤ k ; amt ++ {
        if ( coin[j] ≤ amt)
           dp[amt] += dp[amt - coin[j]]

    }

}

return  dp[K]
```

4. 0-1 Knapsack (Extended)

Given N items with profit V_i and weight W_i. A bag is given with capacity W to carry some objects such that total weight ≤ W and maximize profit in bag.

- Can pick an item only once
- Don't pick fractionally.

row → W+1 cells

$1 <= N <= 500$

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

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

$1 <= W <= 10^9$

$dp[500][10^9]$

↓

5×10^{11}

→ 2×10^9

0 . . . . . N-1

N wt[ ]

W val[ ]

→ int dp[N+1] [W+1]

maxprofit (N, W)

↓

dp[N] [W]

TC : $O(N*W)$

SC : $O(N*W)$ → $O(W)$

$1 <= N <= 500$

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

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

$1 <= W <= 10^9$

× N, W

1 item

↓

50

Max val = 500 × 50

= 25000

= 0.25×10^5

ar → $10^5 - 10^6$

maxProfit $\boxed{N, W}$

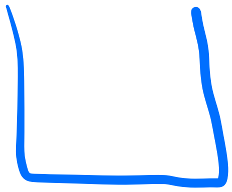
↓

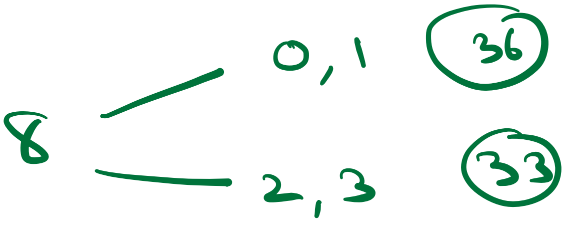
$\boxed{N, value}$

N = 4

	0	1	2	3
wt[i]	3	5	6	2
val[i]	20	16	13	20

W=8





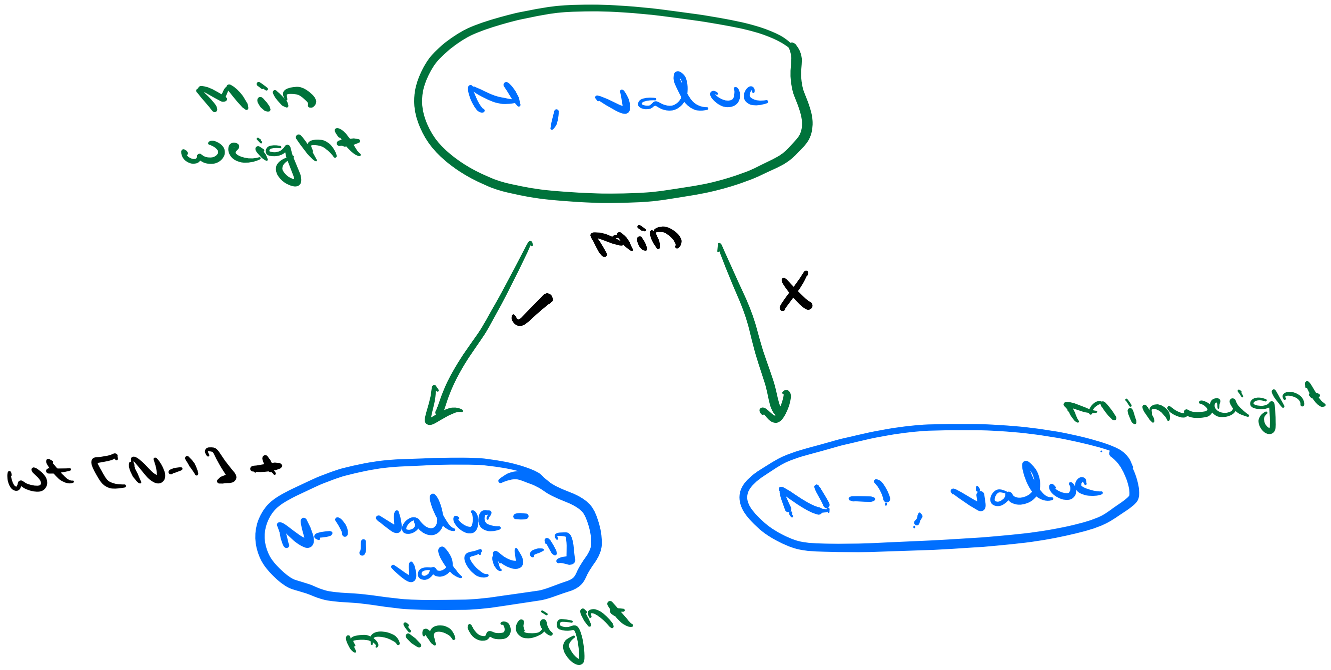
0 1 . . . N-1

budget ← wt

luxury ← val

[illegible]
budget

For luxury, min budget I need?



dp[i][j] = min weight required to get value j from first i items

row → items

col → value

N=3

	0	1	2
val	2	1	3
wt	3	2	4

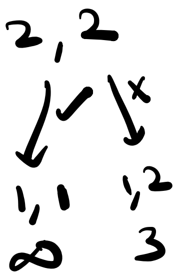
W=7

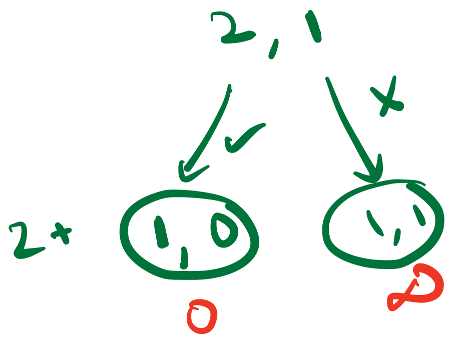
Maxval = 2+1+3 = 6

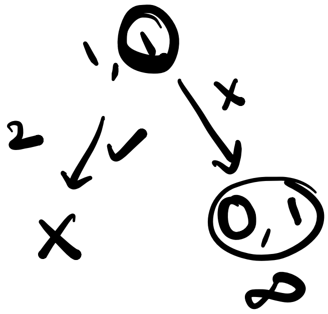
val / luxury

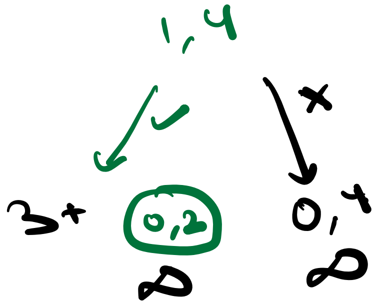
budget ↑

val	wt		0	1	2	3	4	5	6	7
		0	0	∞	∞	∞	∞	∞	∞	∞
2	3	1	0	∞	3	∞	∞	∞	∞	∞
1	2	2	0	2	3	5	∞	∞	∞	∞
3	4	3	0							



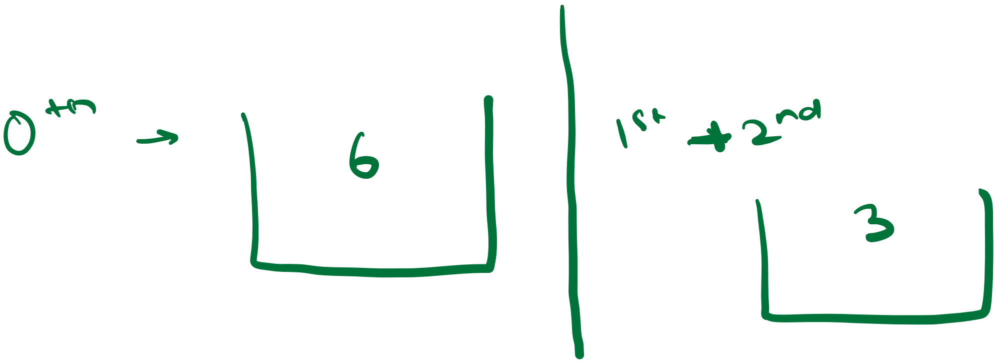


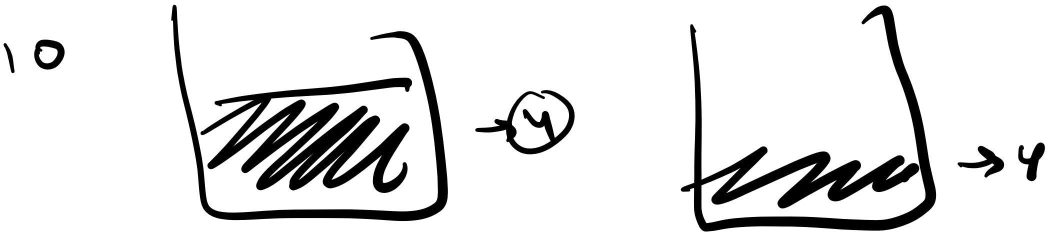




	0	1	2
wt	6	2	1
pf	4	3	1

4





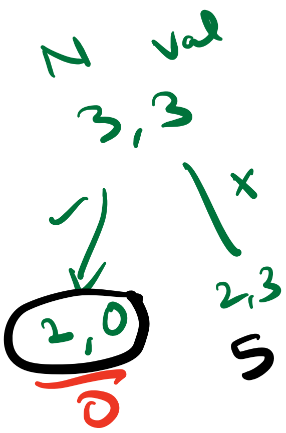
budget ↑

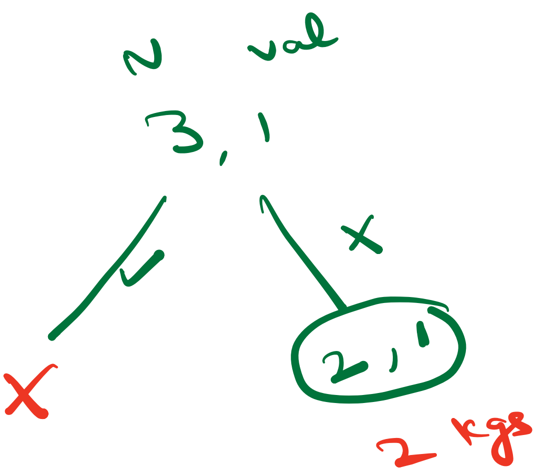
val / luxury

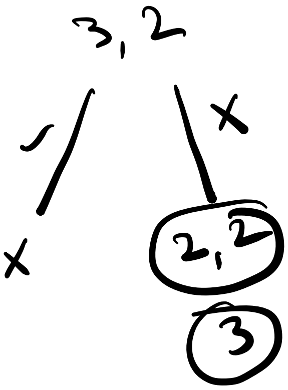
W=7

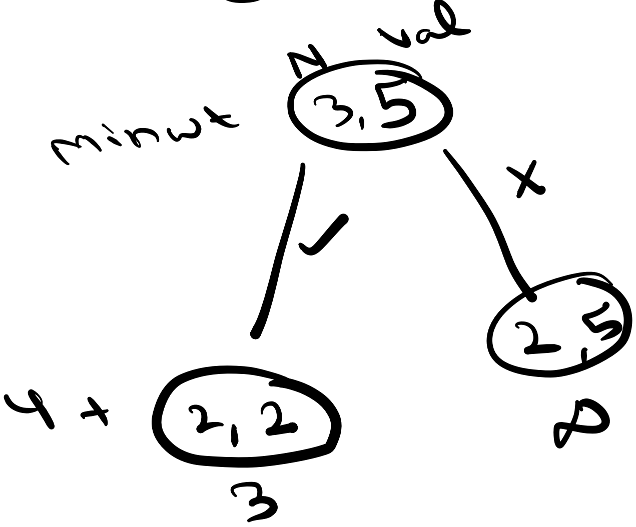
ans=5

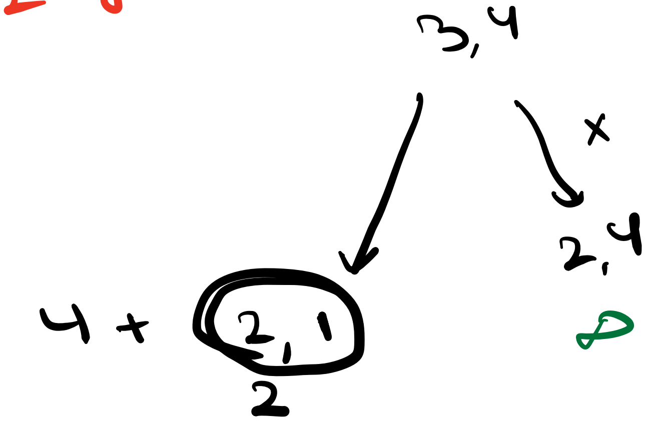
val	wt		0	1	2	3	4	5	6
		0	0	∞	∞	∞	∞	∞	∞
2	3	1	0	∞	3	∞	∞	∞	∞
1	2	2	0	2	3	5	∞	∞	∞
3	4	3	0	2	3	4	6	7	9

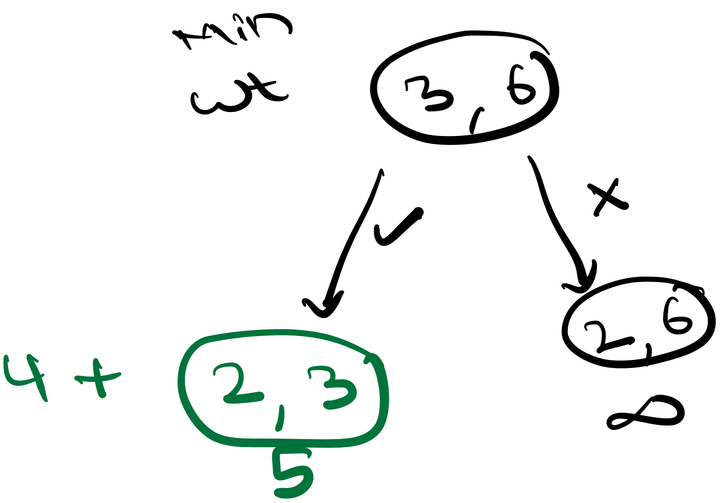








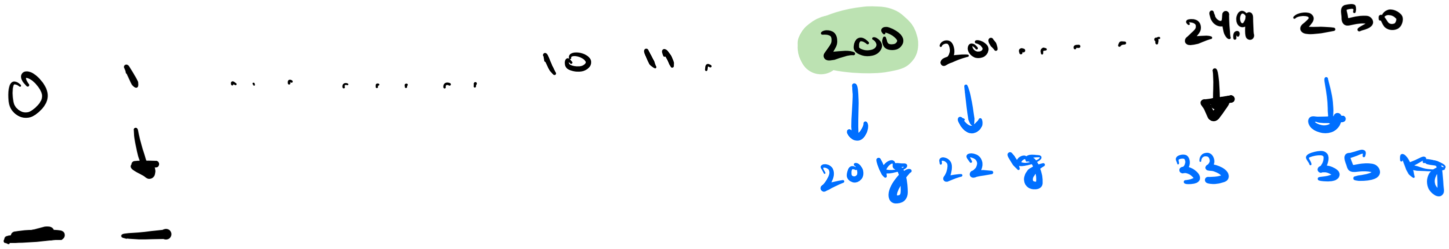




maxVal = sum of val of every item

W=20 kg

maxVal = 250


0
1
10
11
200
201
249
250
20 kg
22 kg
33
35 kg