DP 4

Good Evening

Content

01. Unbounded knapsack
02. Rod cutting problem
03. Coin change permutation
04. Coin change combination
05. 0-1 knapsack 2

Unbounded Knapsack (0-N knapsack)

Given N items each with a weight & value, find max value which can be obtained by picking items such that total weight of all items $\leq K$

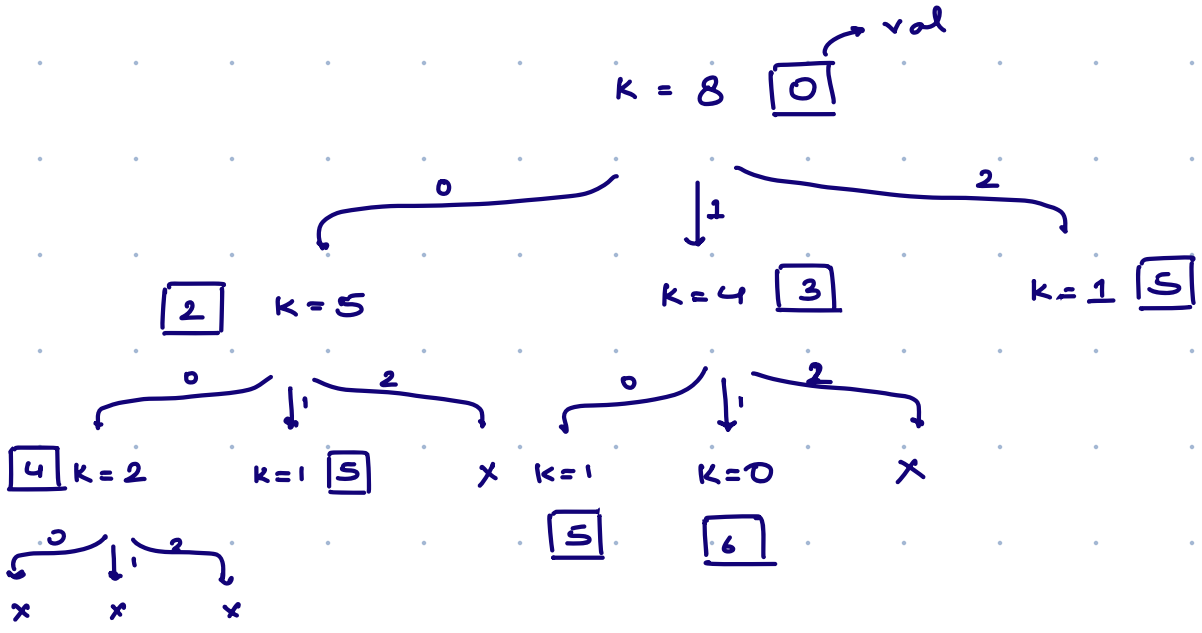
Note 1 :- Every item can be picked infinite no. of times

Note 2 :- We cannot take a part of item

val [] = {2 3 5} (0 1 2)

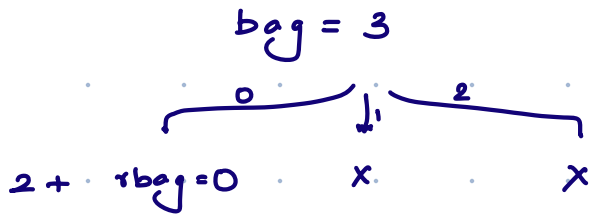
wt [] = {3 4 7}

K = 8



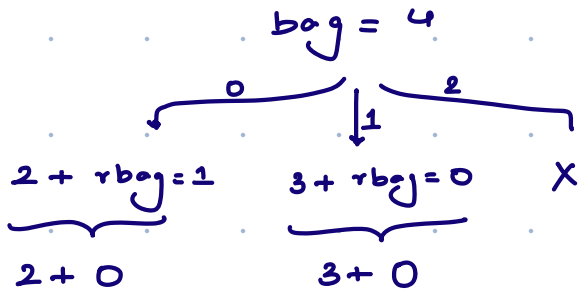
dp[i] = max value which can be generate with i bag

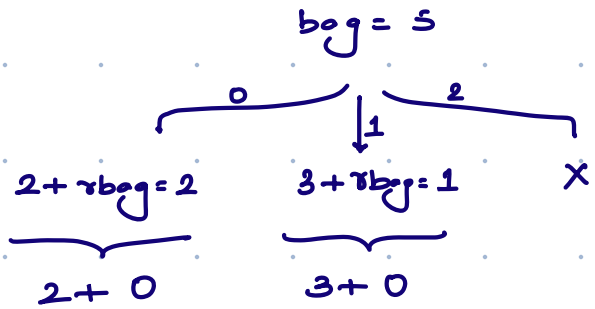
0	1	2	3	4	5	6	7	8
0	0	0	2	3	3	4	5	6

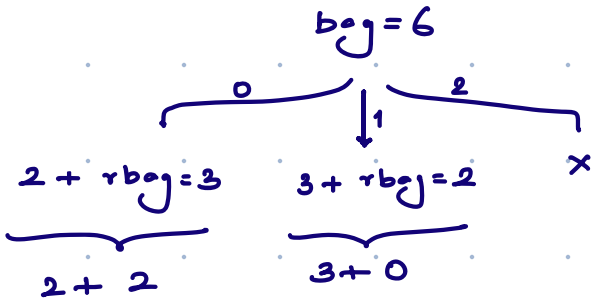


val [] = {2 3 5}
0 1 2
wt [] = {3 4 7}

K = 8








```
int [] dp = new int [k+1];
for (i=0; i<=k; i++){
    for(j=0; j<n; j++){
        if (wt[j] <= i){
            dp[i]= Max( dp[i], val[j] + dp[i- wt[j]);
        }
    }
}
return dp[k];
```

k = bag capacity

TC: O(k*n)

SC: O(k)

Rod cutting

Given a rod of length N & an array of length N+1

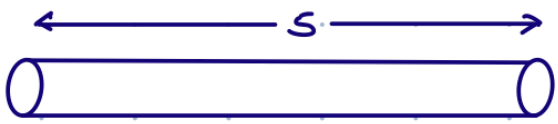
A[i] → price of i length rod.

Find the maximum value we can obtain by selling the rod

Note → we can sell the rod in pieces.

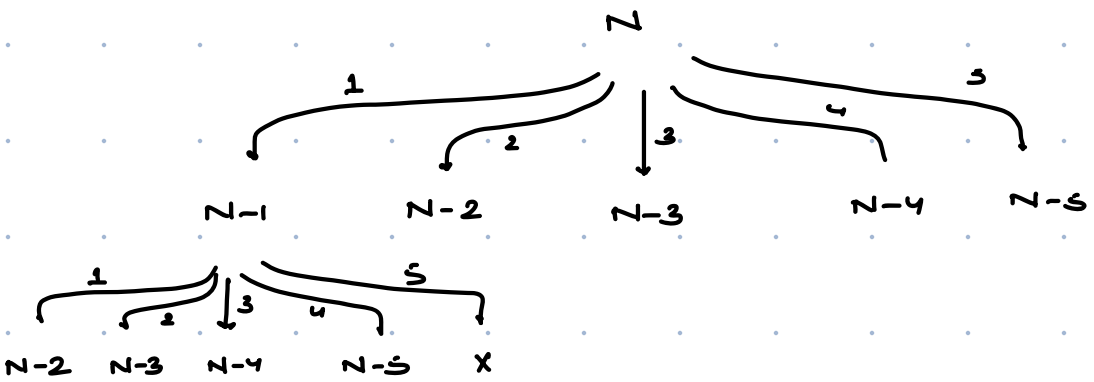
N = 5

A = { 0 1 4 2 5 6 }
0 1 2 3 4 5



rod length	price
1+1+1+1+1	5
1+4	6
2+3	6
5	6
2+2+1	9
1+1+3	4

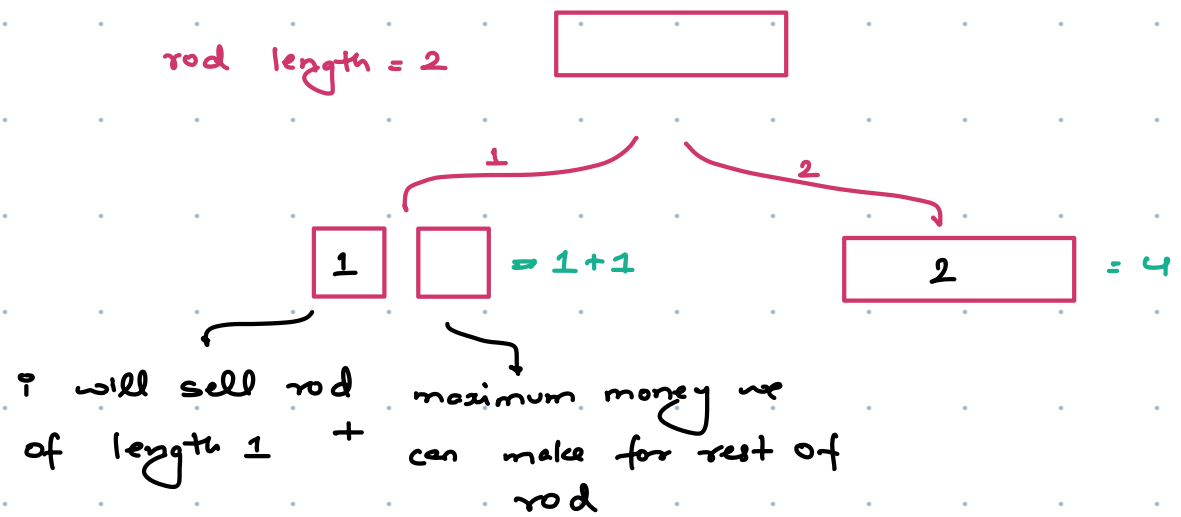
Ans = 9

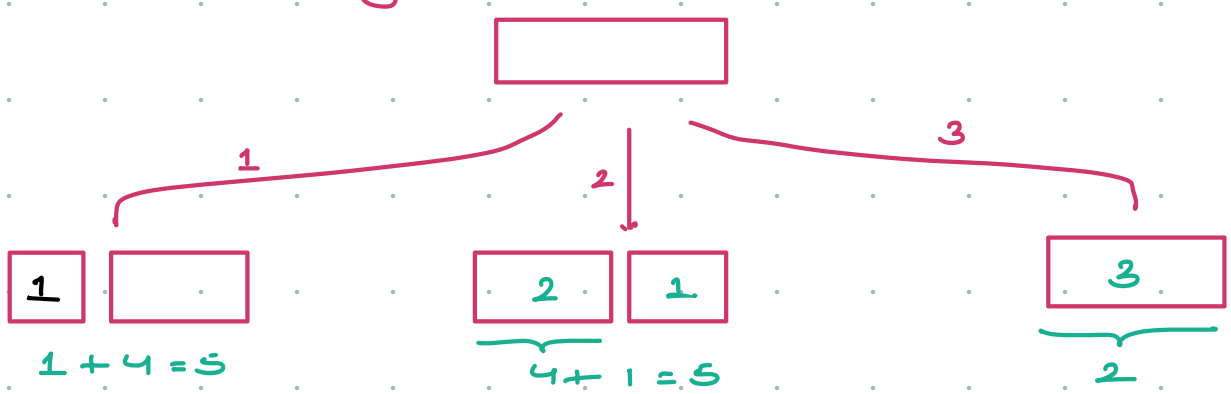


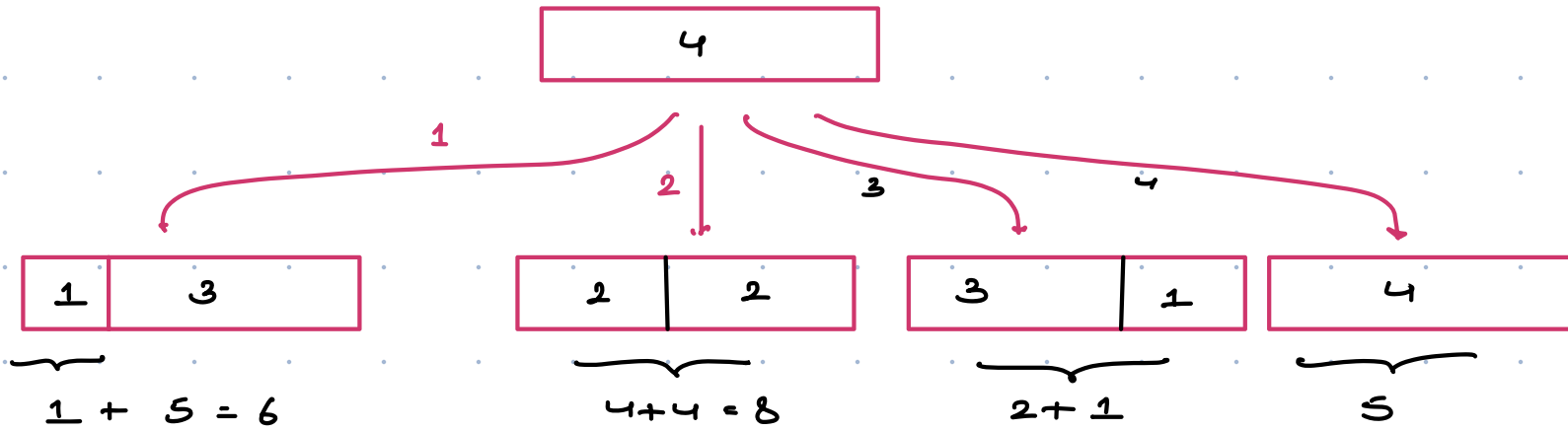
A[ ] = { 0 1 4 2 5 6 }
(0 1 2 3 4 5)

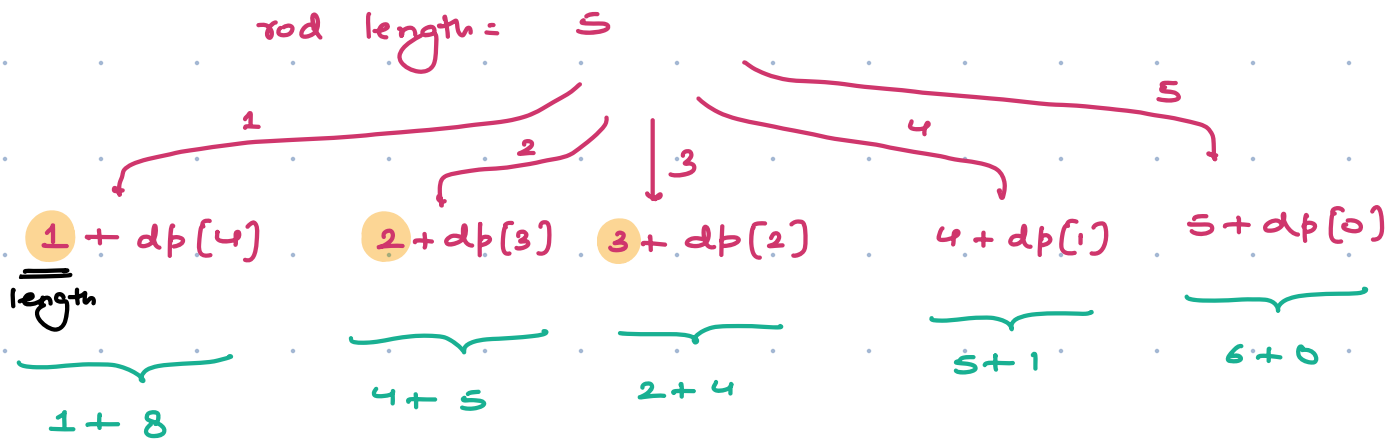
dp[ ] = { 0 1 4 5 8 9 }
(0 1 2 3 4 5)

dp[i] = max value we can generate for i length rod










```
int [ ] dp = new int [n+1];

    dp[0] = 0

for ( i = 1 ; i ≤ n ; i++ ){
|   for( cut = 1 ; cut ≤ i ; cut ++){
|   |
|   |   dp[i] = max ( dp[i] , A[cut] + dp [i-cut] ;
|   |
|   }
|
}
return dp[n];
```

TC: O(n²)
SC: O(n)

Coin change permutation

In how many ways, can sum be equal to N by using coins given in array.

One coin can be used **multiple times.**

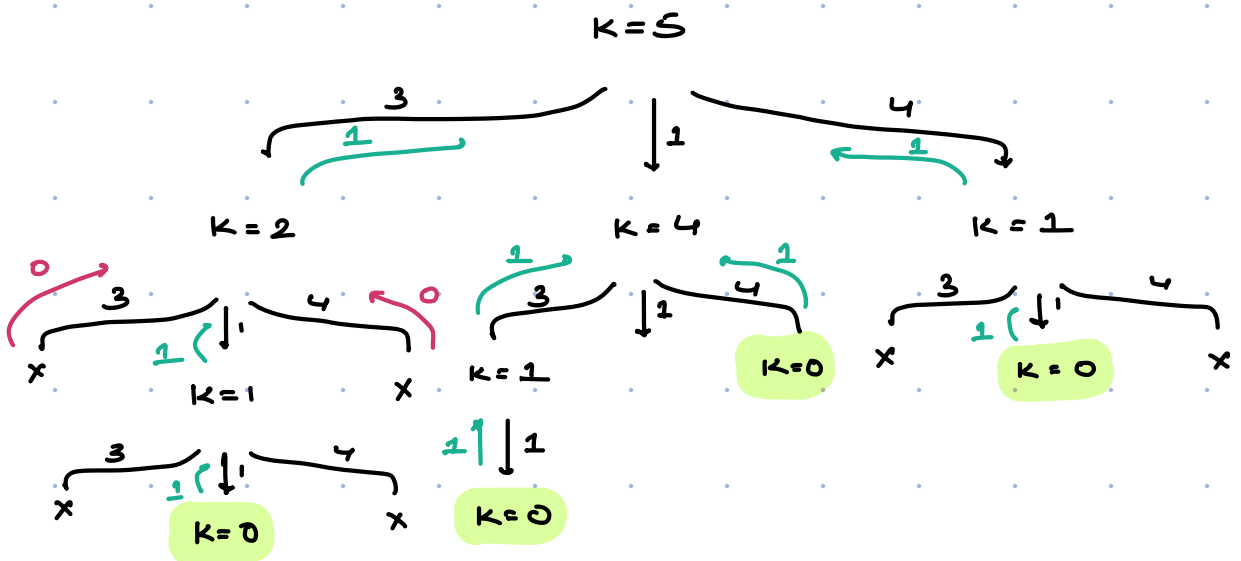
Ordered selection → $(x,y) \neq (y,x)$

$k = 5$

coins[] = {3, 1, 4}

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

Ans = 6

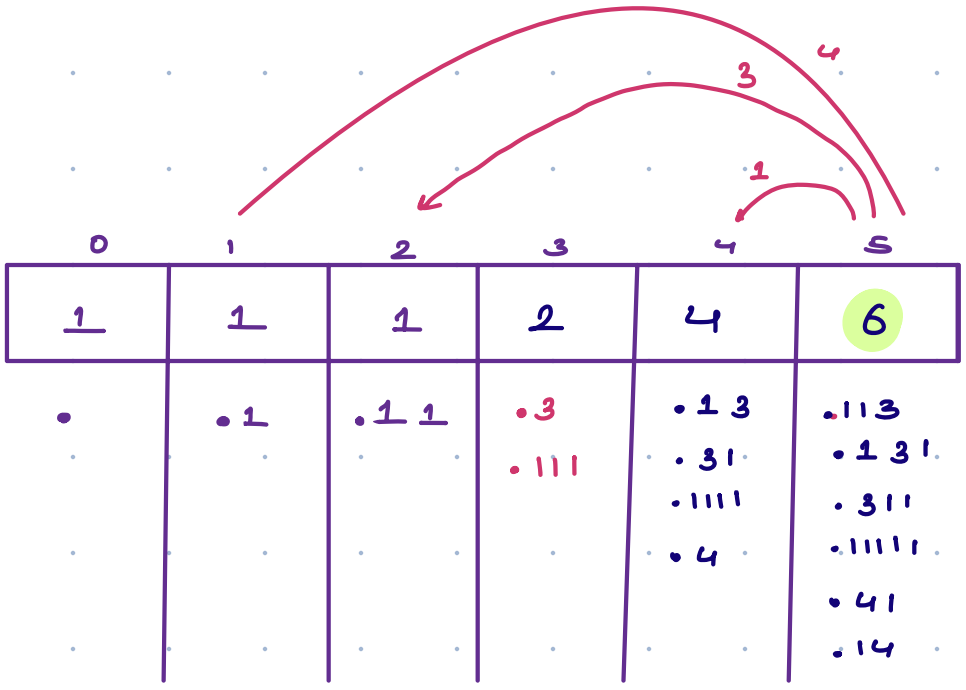


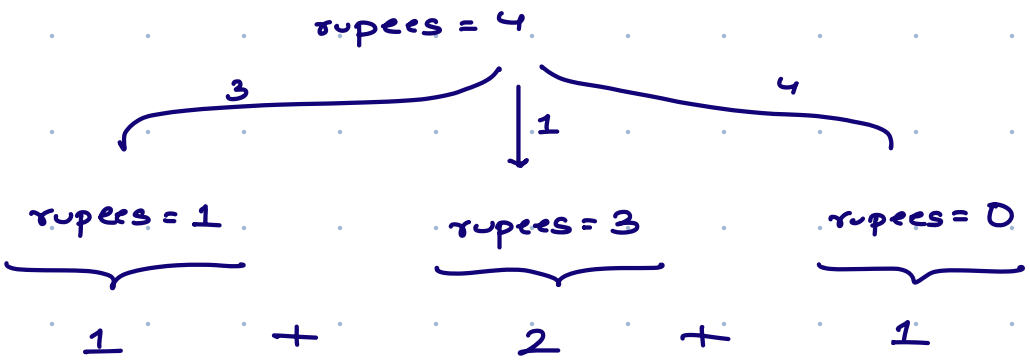
No. of ways to pay rupees = 1 way

Do nothing

dp[i] = No. of ways to pay i rupees

coins = {3, 1, 4}





No. of way = 1 + 2 + 1

```
int [ ] dp = new int [k+1];

dp[0] = 1

for ( i=1 ; i ≤ k ; i++) {
    for ( j=0; j< coins.length ; j++) {
        if ( coins[j] ≤ i) {
            dp[i] += dp[i- coins[j]];
        }
    }
}
```

10:11 pm → 10:21 pm

Coin change combination

In how many ways, can sum be equal to N by using coins given in array.

One coin can be used multiple times.

Unordered selection ⟶ (x,y) = (y,x)

K = 5

coins[] = {3, 1, 4}

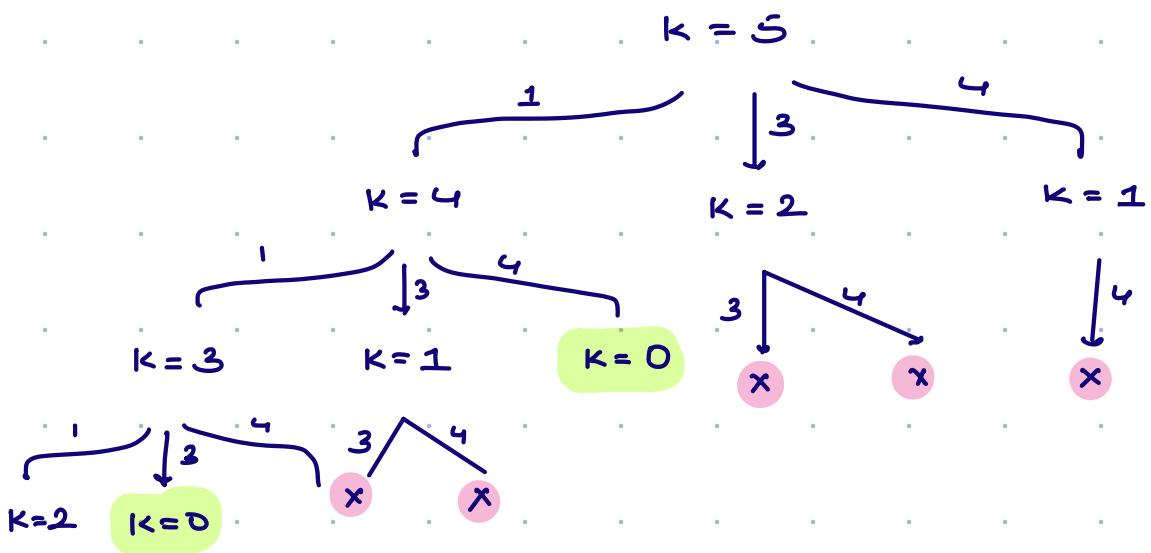
Ans = (1,1,1,1,1) (1,4) (1,1,3)
(4,1) (1,3,1)
(3,1,1)

Arrays.sort(coins)

coins[] = {1, 3, 4}

Ans = 3

Decide how to discard permutation



0	1	2	3	4	5
1	1	1	2	3	3
•	1	11	111	1111	1111
			•3	13	113
				4	14

```
int [] dp = new int [k+1];

dp[0] = 1

for (j=0; j < coins.length; j++){
    for (i=1; i ≤ k; i++){
        if (coins[j] ≤ i){
            dp[i] += dp[i - coins[j]];
        }
    }
}
```

0/1 KnapSack 2

Given N items each with weight & value. Find max value which we can generate by picking items such that total weight of all items ≤ cap

Note → Every item can be picked at max 1 time.

→ we can't go take part of item.

Constraints

$1 \le N \le 500$

$1 \le val[i] \le 50$

$1 \le wt[i] \le 10^9$

$1 \le cap \le 10^9$

0-1 knapsack

SC : $O(N * cap)$

: $O(500 * 10^9)$

: $O(5 * 10^{11})$ ✗

*

Buyer 1 ⇒ 20 lakh

→ Show me all the cars that are under this budget.

Buyer 2 ⇒ 20 lakh

L_1	L_2	L_3	L_4	L_5
15L	20L	19L	25L	30L

Buyer 1 → With bag capacity of 20L, what is maximum value we can generate

Buyer 2 → For all the values, get the minimum weight possibble & then choose the one which fits inside your bag

Buyer 1 → Maximum value we can generate with i elements & j capacity

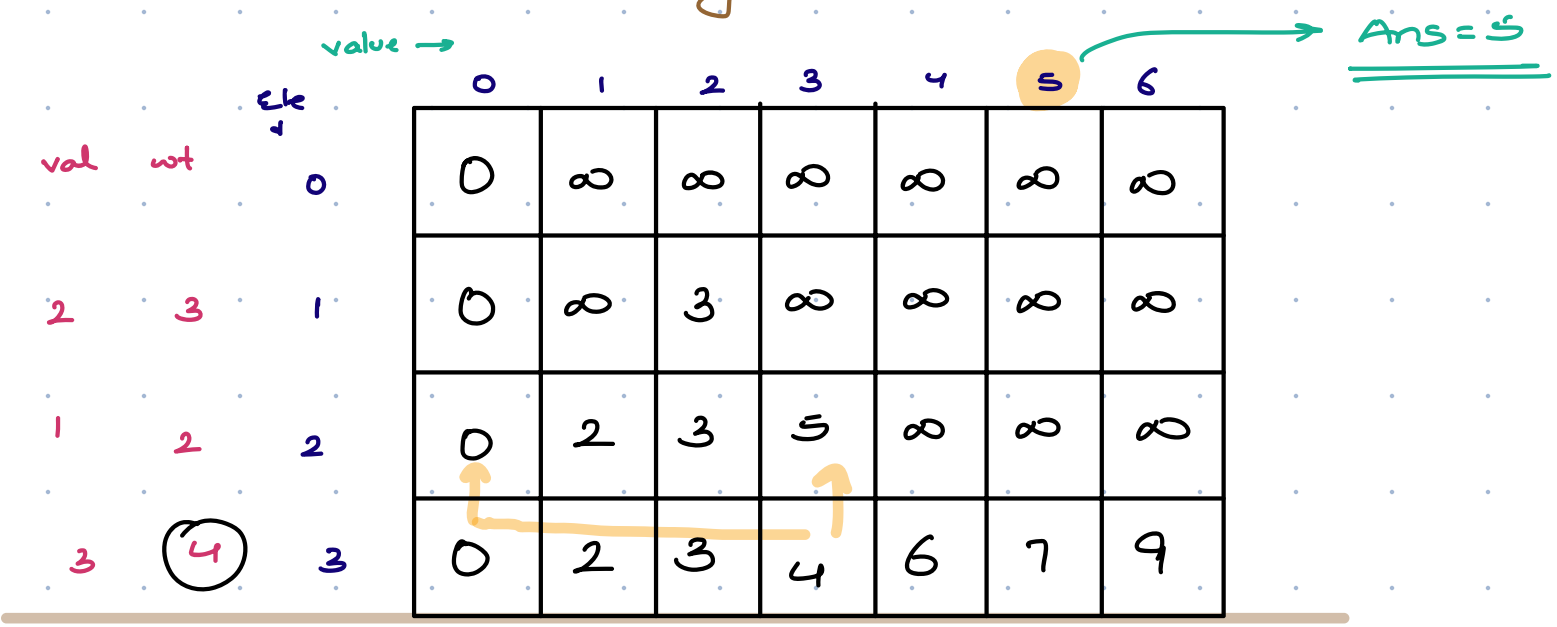
Buyer 2 → Minimum weight required to generate j value with i elements

value[] = { 2 1 3 }

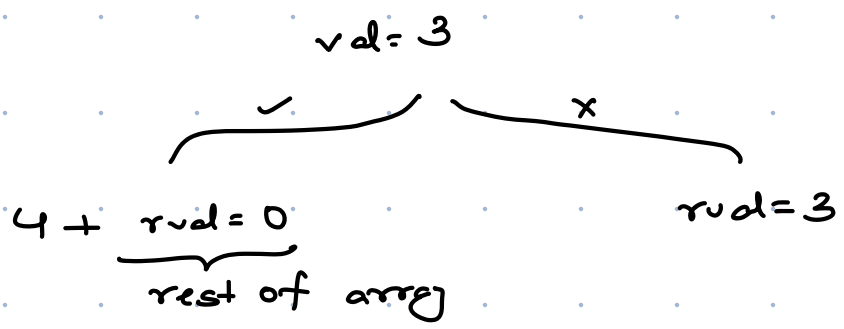
weight[] = { 3 2 4 }

cap = 7

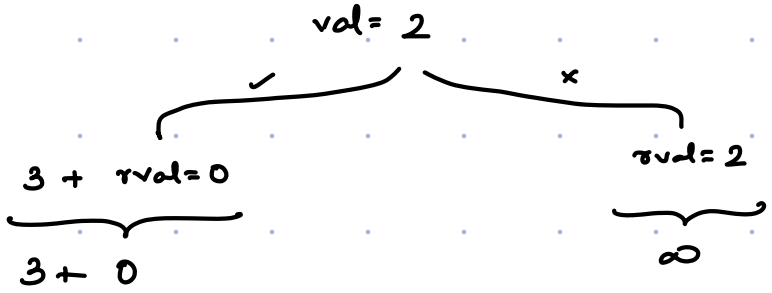
max value generated = 2+1+3 = 6

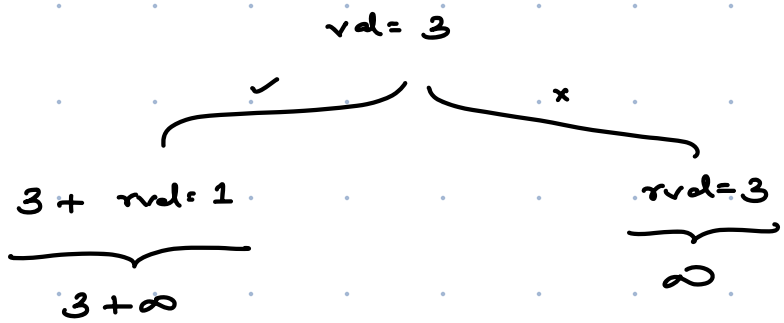


dp[i][j] = minimum weight required to generate j value with i items




4+0
5

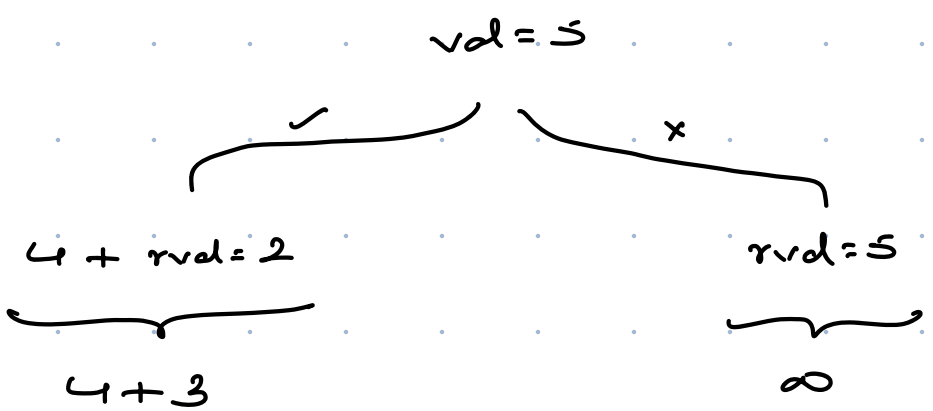

val= 2
✓
x
3 + rval=0
3+ 0
rval= 2
∞

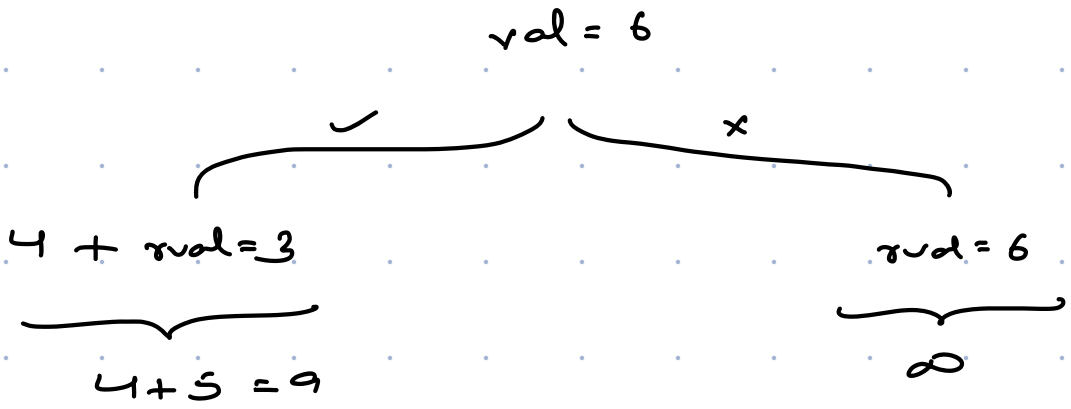

val= 3
✓
x
3 + rval: 1
3+∞
rval=3
∞

✓

val → 4

3


val=5
✓
x
4 + rval=2
4+3
rval=5
∞



SC : O(N * max value)

: O(N * sum of all values)

O(500 * (500 * 50))

: O(125 * 10^5)

: O(1.25 * 10^7)

* find sum of all values

```
sum = 0
for (i=0; i<n; i++){
    sum += val[i];
}
```

```
int [][] dp = new int [n+1][sum+1];

for (i=0; i≤n; i++){

    for (j=0; j < dp[0].length; j++){

        if (j==0)  dp[i][j]=0;

        else if (i==0 && j!=0)  dp[i][j]=∞

        else {

            int npick = dp[i-1][j];
            int pick = ∞
            int rval = j - val[i-1];
            if (rval ≥0 && dp[i-1][rval]!=∞){
                pick = wt[i-1] + dp[i-1][rval];
            }
            dp[i][j]= min(pick, npick);

        }

    }
}

ans=0
for (j=sum; j≥0; j--){
    if (dp[N][j] ≤ k) {ans=j  break};
}
return ans;
```