Q. Knapsack 0/1

Given N items each item with a weight & a value, find max value which can be obtained by picking items such that the total weight of all items <= K

1) Each item can be picked at max once.

2) We can't take a part of the item.

Ex: $N=4$, $K=50$

N :	1	2	3	4
W :	20	10	30	40
V :	100	60	120	150
$\frac{V}{W}$:	5	6	4	3.75

Idea 1 :-

Pick the items based on their value (Greedy)

4^{th} & 2^{nd} $\Rightarrow$ 150+60

$\Rightarrow$ 210. ✗

Idea 2 :- Pick the items based on V/W ratio (Greedy)

2^{nd}, 1^{st} $\Rightarrow$ 60 + 100 = 160. ✗

→ GREEDY is NOT working.

ans: Pick 3^{rd} + pick 1^{st} : 120 + 100 = 220.

Brute force

Generate all possibilities and check the case with weight <= K & max value.

$$TC: O(2^N)$$ { Backtracking Sol^n }

$$SC: O(N)$$

↳ stack size.

Constraints

$$1 <= N <= 10^3$$

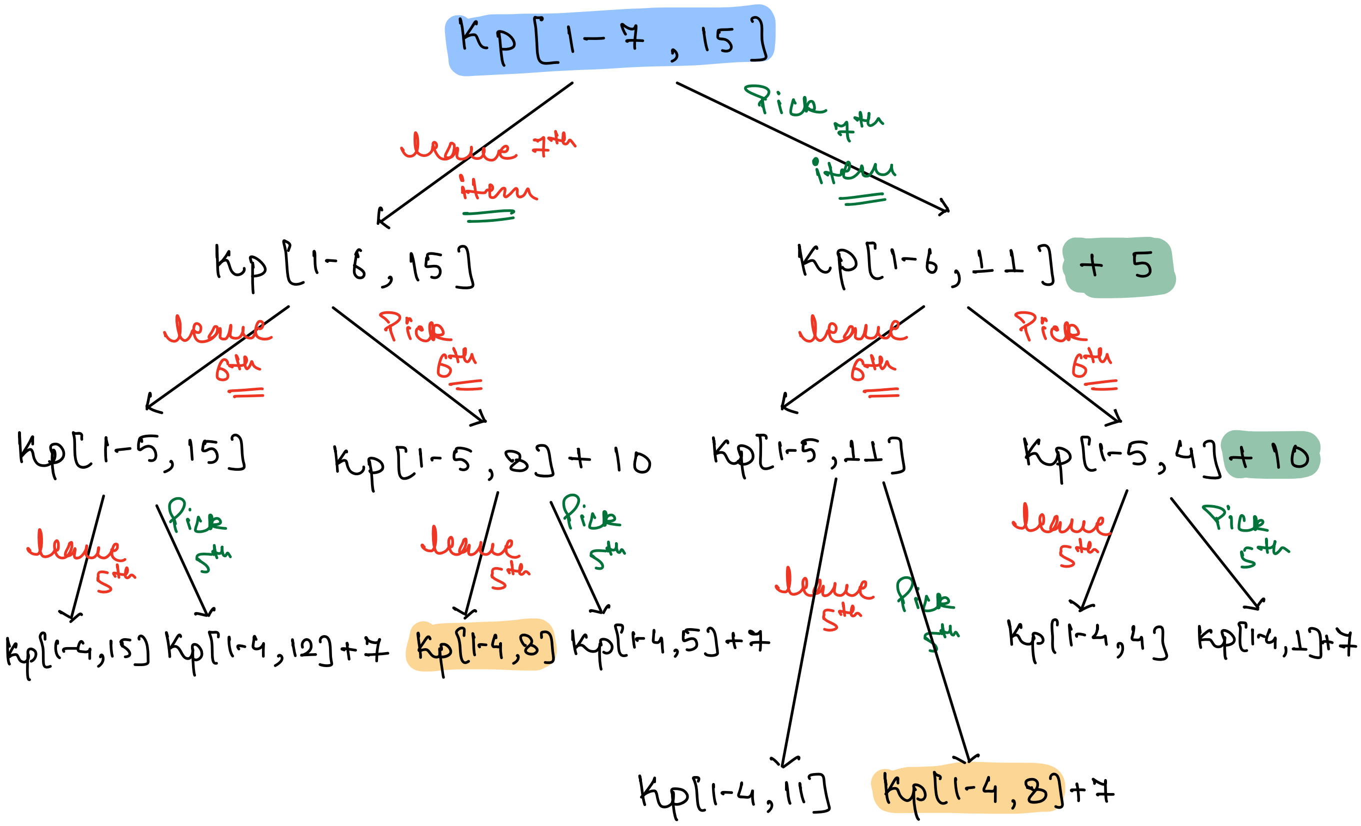
$$1 <= K <= 10^3$$

} 2^{1000} ✗

N=7 K= 15

N:	1	2	3	4	5	6	7
W[]:	4	1	5	4	3	7	4
V[]:	3	2	8	3	7	10	5

$\boxed{\text{Kp[1-7, 15]: Max value which can be obtained using items 1 to 7 with weight <=15}}$



→ Optimal Substructure
→ Overlapping subproblems } DP.

Dp state :

dp[i,j] : max value using [1 to i] items s.t total weight <= j.

Dp Expression

$$dp[i,j] = \text{Max}\begin{cases} \text{leave } i^{th} \text{ item} & \text{Picking } i^{th} \text{ Item} \\ dp[i-1, j] \ , & dp[i-1, \underbrace{j - w[i]}] + v[i] \end{cases}$$

$j - w[i] >= 0$

$j >= w[i]$

Dp table :- ans: dp[N][K]

* int dp[N+1][K+1] = {-1}

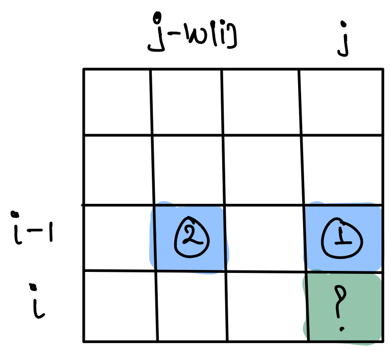
```
int kp(int dp[][], int i, int j, v[], w[]){
    if( i==0 || j==0)
        return 0;
    if( dp[i][j] == -1 ) {
        a = kp(dp, i-1, j, v, w) // leave ith
        if( j >= w[i] ){ //Pick ith item
            a = max(a, kp(dp, i-1, j-w[i], v, w)+v[i])
        }
        dp[i][j] = a;
    }
    return dp[i][j];
}
```

TC: O(N*K)

SC: O(N*K) + stack space.

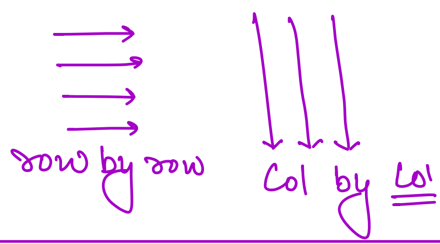
$$dp[i,j] = \text{Max}\left\{ \underbrace{\overset{\text{leave ith item}}{dp[i-1,j]}}_{i>0} \;,\; \overset{\text{Picking ith Item}}{dp[i-1,\underbrace{j-w[i]}_{\substack{j-w[i]>=0 \\ j>=w[i]}}]+v[i]} \right\}$$

```
int Kp(int N, int K, int W[], int V[]){
    int dp[N+1][K+1]
    // Base case
    for(j=0; j<=K; j++)
        dp[0][j] = 0;
    // How to fill the Matrix
```



i) Top to down

ii) Left to Right




```
    for(i=1; i<=N; i++){
        for(j=0; j<=K; j++){
            a = dp[i-1][j]
            if(j>=W[i]){
                a = max(a, dp[i-1][j-W[i]]+V[i])
            }
            dp[i][j] = a;
        }
    }
    return dp[N][K];
}
```

TC: $O(N*K)$

SC: $O(N*K)$

Ex :- $N=5, K=7$

items :	1	2	3	4	5
W[] :	3	6	5	2	4
V[] :	12	20	15	6	10

dp[6][8]

W		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
3	1	0	0	0	12	12	12	12	12
6	2	0	0	0	12	12	12	20	20
5	3	0	0	0	12	12	15	20	20
2	4	0	0	6	12	12	18	20	21
4	5	0	0	6	12	12	18	20	22

5th

5,7

dp[2,6] = man(dp[1,6] , dp[1,0] + 20)

dp[2,7] = max (dp[1,7] , dp [1,1] + 20)

dp[3,3] = max (dp[2,3] , x)

dp[4,2] = max (dp[3,2] , dp[3,0] + 6)

dp[4,3] = max (dp[3][3] , dp[3][1] + 6)

dp[4,5] = max (dp[3][5] , dp[3,3] + 6)

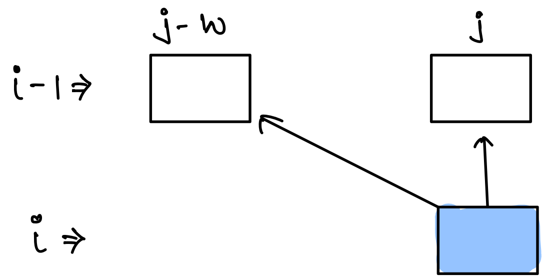
15 , 12+6

dp[5,6] = max (dp[4,6] , dp [4,2] + 10)

20 , 16

dp[5,7] = max (dp[4,7] , dp[4,3] + 10)

(21 , 22)



$$dp[i][j] = \max(dp[i-1,j],\ dp[i-1,j-w] + V[i])$$

$(j >= w)$

$$dp[5,7] = \max(\underbrace{dp[4][7]}_{21},\ \underbrace{dp[4][7-4]}_{12} + 10)$$

$$dp[1,3] = \max(dp[0,3],\ \underbrace{dp[0,3-3]}_{dp[0,0]} + 12)$$

⇒
```
i = N, j = K
while (i > 0 && j > 0) {
    if (dp[i][j] == dp[i-1][j])   // Not picking ith item
        i = i-1;
    else {
        // ith element is present in ans.
        ans.add(i);
        i = i-1
        j = j-W[i];
    }
}
```

Space Optimization :-

→ At any given time we only need 2 rows.

0^{th}	~~0^{th}~~ 2^{nd}
1^{st}	~~1^{st}~~ 3^{rd}

dp[2][K+1]

0^{th}	$\longrightarrow$	0^{th}
1^{st}	$\longrightarrow$	1^{st}
2^{nd}	$\longrightarrow$	0^{th}
3^{rd}	$\longrightarrow$	1^{st}
4^{th}	$\longrightarrow$	0^{th}
5^{th}	$\longrightarrow$	1^{st}
	⋮	
i^{th}	$\longrightarrow$	$i \% 2$
$(i-1)^{st}$	$\longrightarrow$	$(i-1) \% 2$
n	$\longrightarrow$	$n \% 2$

Disadvantage :-

→ We won't be able to trace back the ans.

$$\boxed{\begin{array}{l} i \Rightarrow i \% 2 \\ i-1 \Rightarrow (i-1) \% 2 \end{array}}$$

TC : O(N*K)

SC : O(K)

return dp[N%2][K]

B.F	→	DP Recursive	→	DP Iterative	→	DP iter. with Space optimization
TC: $O(2^N)$		TC: $O(NK)$		TC: $O(NK)$		TC: $O(NK)$
SC: $O(N)$		SC: $O(NK)$ + Stack Space		SC: $O(NK)$		SC: $O(K)$

Q. Exactly same as previous problem.

→ A single item can be picked as many times as we want? (∞ Knapsack)

$N = 4$

	1	2	3	4
w[] :	20	13	10	40
v[] :	100	66	40	150

$K = 50$

$$\underbrace{dp[i][j]}_{\text{max value using [1 to i] items s.t total wt} \le K} = \max\big(\underbrace{dp[i-1][j]}_{\text{leave } i^{th}},\ \underbrace{dp[i][j-w[i]]+v[i]}_{\text{pick } i^{th}}\big)$$

max value using [1 to i] items s.t total wt <= K.

i, j

i-1, j   i, j-w[i]

i-1, j-w[i]   i, j-2w[i]

i-1, j-2w[i]   i, j-3w[i]