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 items based on the value.

Pick 4^{th} & 2^{nd}

$V = 150 + 60 = 210$ ✗

⇒ Greedy based on Value.

Idea 2: Greedy based on $\frac{V}{W}$ ratio.

Pick 2, 1

$V = 60 + 100 \Rightarrow 160$ ✗

Ans: Pick 1, 3

$V = 100 + 120 = 220.$

⇒ Generate all the subsets and find the subset with max Value with sum $<= K$.

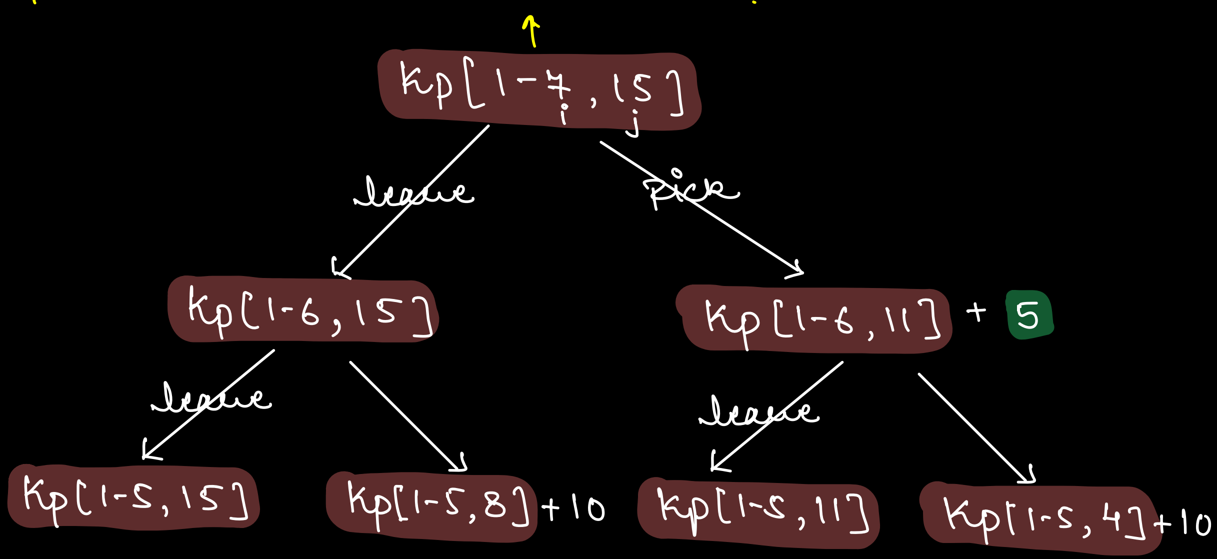
$$TC: O(2^N)$$
$$SC: O(N)$$

* Constraints :- $1 =< N <= 10^3$
$1 =< K <= 10^3$

N = 7	1	2	3	4	5	6	7
W	4	1	5	4	3	7	4
V	3	2	8	3	7	10	5

K = 15

Max value that can be obtained from the products 1 to 7 with $W <= 15$.



→ Optimal Substructure
→ Overlapping subproblems } DP ✓

dp State

dp[i, j] : Max value using [1-i] with wt <= j.

dp Expression :

$$dp[i,j] = \max\{\underbrace{dp[i-1, j]}_{\text{leave ith}}, \underbrace{dp[i-1, j-w[i]] + v[i]}_{\text{Pick ith}}\}$$

⇒ $j - w[i] >= 0$

⇒ $j >= w[i]$

dp table :

Final Ans : return dp[N][K]

dp[N+1][K+1]

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

int Kp(int dp[][], N, K, wt[], v[], i, j){
    if(i==0 || j==0) return 0;

    if(dp[i][j] == -1){
        // Leave ith item.
        int a = kp(dp, N, K, wt, v, i-1, j);
        if(j >= w[i]){
            a = max(kp(dp, N, K, wt, v, i-1, j-w[i])+v[i],
                a)
        }                              pick ith item.
        dp[i][j] = a;
    }
    return dp[i][j];
}

main(){
    int dp[N+1][K+1] = {-1};
    return Kp(dp, N, K, wt, v, N, K);
}
```

TC: O(NK)

SC: O(NK)

Iterative Code

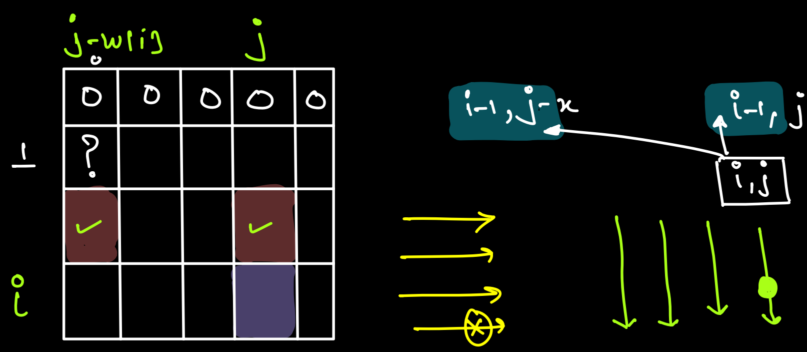
$$dp[i,j] = \max\{dp[i-1,j],\ dp[i-1,\ \underbrace{j-w[i]}_{j >= w[i]}] + v[i]\}$$

⇒ Base Case

$i = 0$ ✓

$j = 0$ ✗

```
int kpIterative(wt[], v[], N, K) {
    int dp[N+1][K+1];
    // Base Case.
    for(j = 0; j <= K; j++) dp[0][j] = 0;
```




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

}

return dp[N][K];

}

$$\boxed{\begin{array}{l} TC: O(NK) \\ SC: O(NK) \end{array}}$$

N=5 :	1	2	3	4	5
Wt[] :	3	6	5	2	4
V[] :	12	20	15	6	10

K = 7

dp[6][8]

wt		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

$\boxed{i-1, j-w}$ $\boxed{i-1, j}$

$\boxed{i, j}$

$i, j \Rightarrow i-1, j-w[i]$

5,7
↓
4,7-4
↓
3,3
↓
2,3
↓
1,3
↓
0,3-3 = 0,0

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

$$dp[1,3] = \max\left(dp[0][3],\ dp[0]\underbrace{[3-3]}_{3 \geq 3} + v[1]\right)$$

$$= \max\left(0,\ \underbrace{dp[0][0]}_{0} + \underbrace{v[1]}_{12}\right)$$

$$= \max(0, 12) = \underline{\underline{12}}$$

$$dp[\underset{i}{2}, \underset{j}{6}] = \max\left(dp[1][6],\ dp[1][6-6] + v[2]\right)$$

$$6 \geq 6$$

$$j \geq w[2]$$

$$= \max(12,\ 0+20) = \underline{\underline{20}}$$

$$dp[\underline{4,2}] = \max\left(\underset{\downarrow}{\underset{0}{dp[3,2]}},\ \underset{\downarrow}{\underset{0}{dp[4-1, 2-2]}} + \underset{\downarrow}{\underset{6}{v[4]}}\right)$$

$$2 \geq w[4]$$

$$\underline{\underline{2}} \geq 2$$

$$= \max(0, 6) = 6$$

$$dp[\underset{i}{4}, \underset{j}{5}] = \max\left(dp[3,5],\ \underbrace{dp[3, 5-2]}_{12} + \underbrace{v[2]}_{6}\right)$$

$$5 \geq w[4] \checkmark$$

$$= \max(15, 18) = \underline{\underline{18}}$$

$$dp[5,7] = \max\left(dp[4,7],\ dp[4, 7-\underbrace{w[5]}_{4}] + \underbrace{v[5]}_{10}\right)$$

$$= \max\left(21,\ \underbrace{dp[4,3]}_{12} + 10\right) = \underline{\underline{22}}$$

⇒

```
i = N, j = K
list<int> ans;
while ( i>0 && j>0 ) {
        if (dp[i, j] == dp[i-1, j]) {
                i--;
        }
        else {
            // pick ith product
            ans.add(i);
            i = i-1 , j = j-w[i];
        }
}
```

\# Can we Optimise the SC ?

dp[2, K+1]

0	0th
1	1st

row no →	index
0	0th
1	1st
2	0th
3	1st
4	0th
⋮	⋮

i^{th} row $\Rightarrow$ $i \% 2$

```
int KpIterative ( wt[] , v[] , N , K ) {
    int dp[2][K+1];
    //Base Case.
    for( j= 0; j<=K; j++) dp[0][j] = 0;

    for( i= 1; i<=N; i++ ) {
        for( j = 0; j<=K; j++) {
            a = dp[(i-1)%2][j];
            if( j >= w[i] ) {   j-w[i] >= 0 ✓
                a = max(a, dp[(i-1)%2][j-w[i]] + v[i]);
            }
            dp[i%2][j] = a;
        }
    }
    return dp[N%2][K];
}
```

Disadvantage: We won't be able to trace to the items picked in the ans.

TC: $O(NK)$

SC: $O(K)$

$9^{th} \Rightarrow 8^{th}$

$9 \% 2$ $\Downarrow$ 1st

$8 \% 2$ $\Downarrow$ 0

we can also take 1D arr and use modulous operation to store 2 values in same row

Q. Exactly same as above problem.

N items, K is max wt.

⇒ A single item can be picked any no. of times.

(∞ Knapsack)

$$dp[i,j] : \max\left(dp[i-1,j]\ ,\ dp[i]\underbrace{[j-w[i]]}_{j>=w[i]} + v[i]\right)$$

⇒ Draw the recursive tree for ∞ knapsack.

—— * ——