Today's Content:

→ knap sack → (0/1), → (0/∞),

Knap Sack:

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

Note1 Every item can be atmax picked only 1 time ($x = 1$)

Note2 We cannot take a part of item.

Ex: N = 4 items, k = 50

N =	1	2	3	4	ans = Pick 1 & Pick 3, ans = 220
w[] =	20	10	30	40	
v[] =	100	60	120	150	
v/w =	5	6	4	3.75	

Ideal:

x → Take elements in max value : { 150, 40 | 120, 30 | 100, 20 | 60, 10 } ans = 210

// Both greedy approaches fails

x → Take element in of v/w ratio : { 6, 60, 10 ← 5, 100, 20 | 4, 120, 30 | 3.75, 150, 40 } ans = 160

Idea2: Get all Subsets (2^N) weight $\leq k$ & get max value

↓

optimize ?

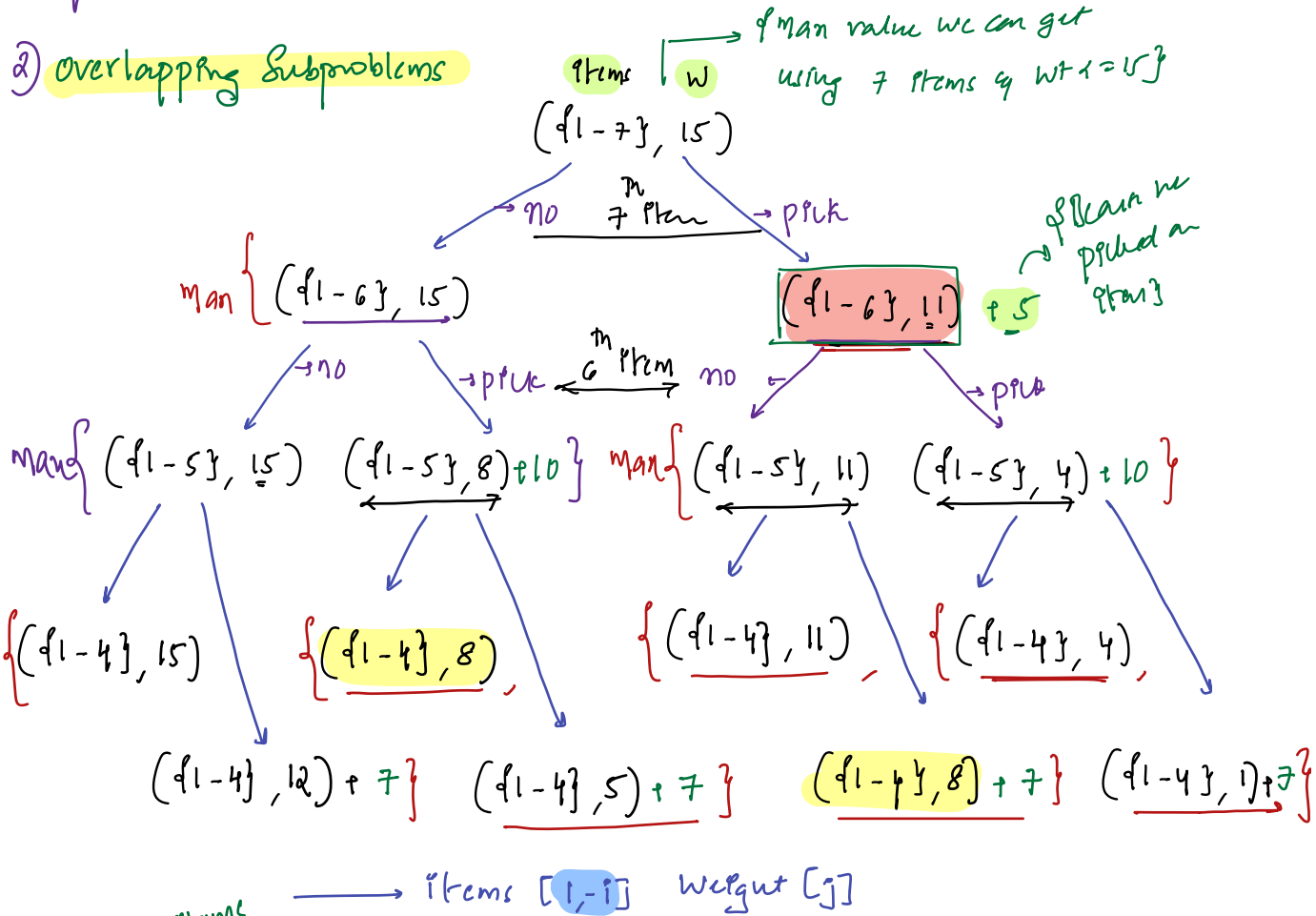
N=7	1	2	3	4	5	6	7
W[]	4	1	5	4	3	7	4
V[]	3	2	8	3	7	10	5

K = 15

Max weight we can pick

1) optimal substructure

2) overlapping subproblems



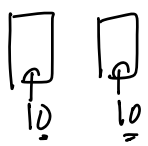
→ items [1-i] weight [j]

// dp[i][j] = Max value from [1 i] items such that total weight ≤ j

items
weight

// dp[i][j] = max { dp[i-1][j] , dp[i-1][j-w[i]] + v[i] }

if (j ≥ w[i]) : Only then we can pick ith element



// Table: N k

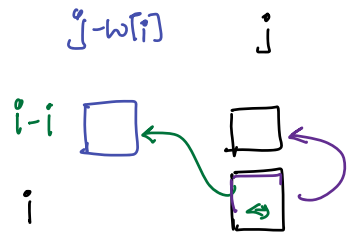
dp[N+1][K+1]

i==0: 0 items :

// Edge Cases: if (i==0) fails : { k ∀ j=0 dp[0][j] = 0 }

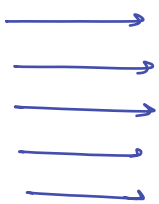
if (j==0) fails : { j==0: no weight : N ∀ i=0 dp[i][0] = 0 }

// Iterative Code : How to fill table?



a) top → down

b) Left → right



// Code :

TL: (N+1)(K+1) * {O(1)}

TC: O(N*K)

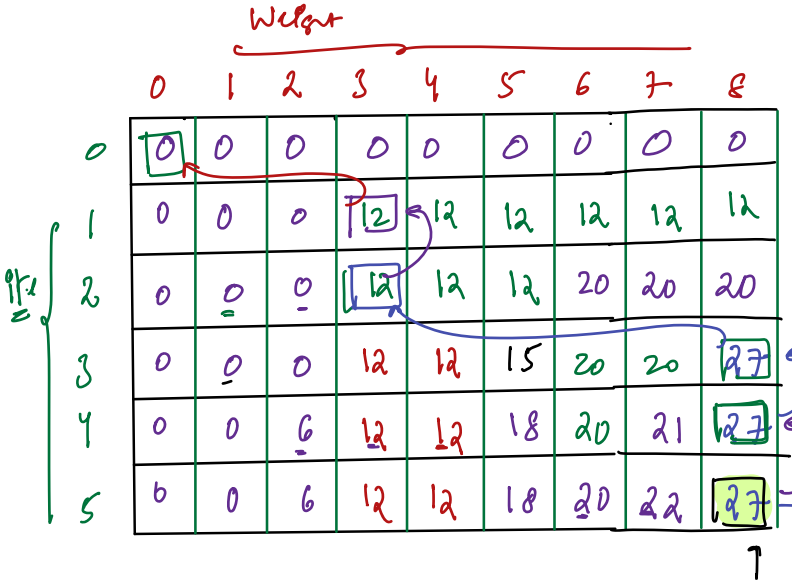
SC: O(N*K) → space optimization → ? {In coming Session}

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

// Tracing:

N = 5, K = 8 :

items:	1	2	3	4	5
w[] :	3	6	5	2	4
v[] :	12	20	15	6	10



i = 5, j = 8 — i: 4, j: 8

dp[5][8] == dp[4][8] — i: 3, j: 8

dp[4][8] == dp[3][8] — i: 2, j: 3

dp[3][8] = dp[2][3] + 15 — i: 1, j: 3
(3rd is picked)

dp[2][3] == dp[1][3] — i: 0, j: 0

dp[1][3] = dp[0][0] + 12
(1st is picked)

→

Finding items picked :

dp[i][j] = max { dp[i-1][j] : { We are not picking i^{th} elem }

dp[i-1][j-w[i]] + v[i] : { picking i^{th} elem }

```
vector<int> ans;
i = N, j = K;
while( i>=1 && j>=1 ){
    if( dp[i][j] == dp[i-1][j] ){
        // we are not
        i = i-1;
    }
    else{
        // we are picking ith elem
        ans.insert(i)
        i = i-1, j = j-w[i]
    }
}
```

→ which elems are picked?

2(8) Exactly same above Problem

Note: A single item can be picked as many times as we want !

(pick items to get max val such that overall weight ≤ k)

Ex:

N =	1	2	3	4
W[i] =	20	13	10	40
V[i] =	100	66	40	150

k = 50 : 200, 190, 240, 238

N = 6

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

k = 15 : max weight

1) optimal substn

2) overlapping

(6, 15)

→ pick

max { (5, 15) , (6, 11) + 7 }

↳ [1 - i] items, j weight

dp[i][j] = max value using [1 - i] total weight ≤ j, item can be used more than once

dp[i][j] = max { dp[i-1][j], dp[i][j - w[i]] + v[i] if (j >= w[i]) }

TODO

Student :

{Tier - 3} ECE Stream :

College Batch ⇒ Fcd 2021 → { Basic language }

4/5 → per week

6 : DSA + Ad DSA

Computer : (CN / DBMS / OS / OOPS) : 1 month

All assignment / 2→3 times

google last | goldman sachs 25 | Amazon 40 | → 6

→ 1 yr

// → 3.5 → {18} Hiring / Clc → : 1y

Her joined paid :

18 : 5 yr

// 6 — lohs —) : { Session + → { Ans thw } + Bch + }

Sacrific Means Success