Knapsack

N objects → Value
N objects → Weight

Value [N] ⇒ Value [i] ⇒ value of the i^{th} object
Weight [N] ⇒ Weight [i] ⇒ Weight of the " "

Bag with a capacity C
(Sum of weights of objects in the bag $\leq$ C)

Select objects s.t. the sum of values is either maximized or minimized.

0/1 KnapSack (Bounded Knapsack)

Q. Toy Shop

N Toys ⇒ Happiness ⇒ {1, 3, 5, 6}
Weight ⇒ {2, 3, 4, 5}

Bag Capacity (C) = 7 Kg.

Max happiness you can put in the Bag.

Sol^n

Greedy is not the solution.

⇒ We need to check for all possibilities.

	0	1	2	3
H =	1	3	5	6
W =	2	3	4	5

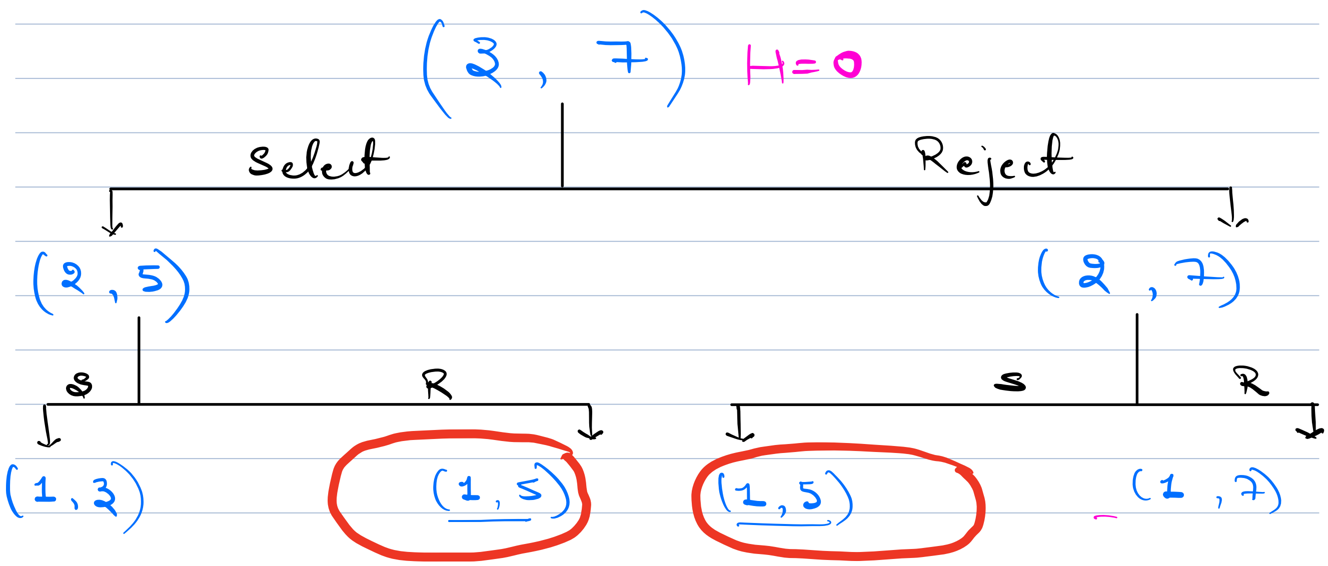
$C = 7$

8 ~~9~~

N objects → Select / Not Select

(idx , capacity left) → Happiness

	0	1	2	3
H =	6	5	3	1
W =	5	4	2	2



State $\Rightarrow$ (index, capacity left) $\rightarrow$ Max Happiness

MaxHappiness (index, capacity)

↳ The max happiness that you can get by deciding for all objects for (0 to idx) with left capacity.

index

↑

$DP[i][j]$ — Reject → $DP[i-1][j]$

— Select. → $DP[i-1][j-W[i]] + H[i]$

↓

remaining capacity

W=3

	0	1	2	3	4	5	6	7
→ 0	0	0	0	0	0	0	0	0
1	0	—	—	—				
2	0				×			
3	0				×			
4	0							

Index

→ Max Hap C=7

$DP[3][7]$

↓

Max Happiness

Toys [0, 3]

C = 7

N

Max H

(N)

C = 6

Ans

$DP[N+1][C+1] = \{0\}$

for (i=1; i <= N; i++) {

for (j=1; j <= C; j++) {

→ if (W[i] <= j) {

DP[i][j] = max((DP[i-1][j]),

(DP[i-1][j - W[i-1]] + H[i-1]))

}

→ else {

DP[i][j] = DP[i-1][j]

}

}

}

return DP[N][C];

T.C. = O(N × C)

Unbounded KnapSack (0-N)

↳ Select an object any no. of times.

0/1 — Reject / Select > Element is processed.

0/N — Reject → Element is processed.

0/N — Select → Not processed as we can select it again.

index

↑

$DP[i][j]$

↓

capacity

Reject → $DP[i-1][j]$

Select → $H[i] + DP[i][j - wt[i]]$

T.C. = $O(N \times C)$

S.C. = $O(N \times C)$ → 2D DP

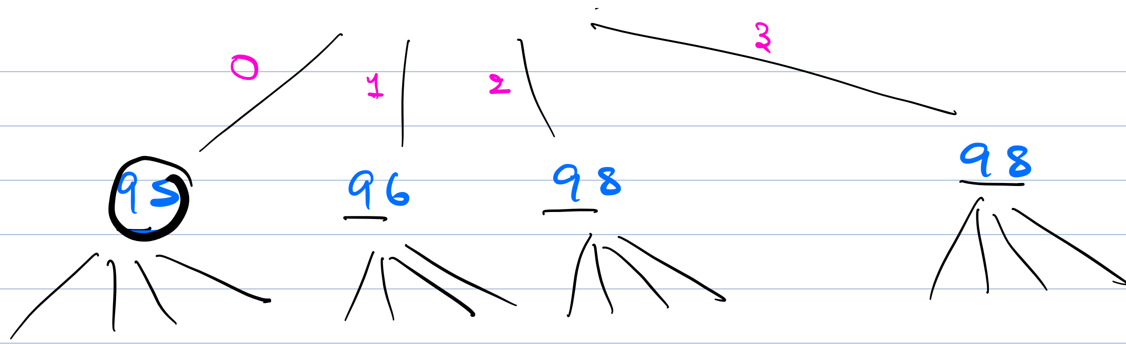
Q. Can we define a 1D DP state for this problem.

index, capacity ~~Happiness~~

Bag = C (N)

	0	1	2	3
H =	6	5	3	1
W =	5	4	2	2

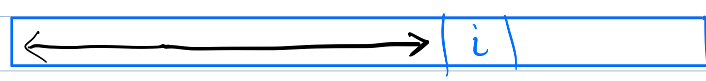
(C = 100)



EOC → N objects

DP[i] → Max happiness in the bag of capacity i.

$$DP[i] = \max\left(\forall_{objects}\left(H[i] + DP[i - wt[obj]]\right)\right)$$



$i = 0$ (capacity = 0)

$H = 0$ ($DP[0] = 0$)

Code

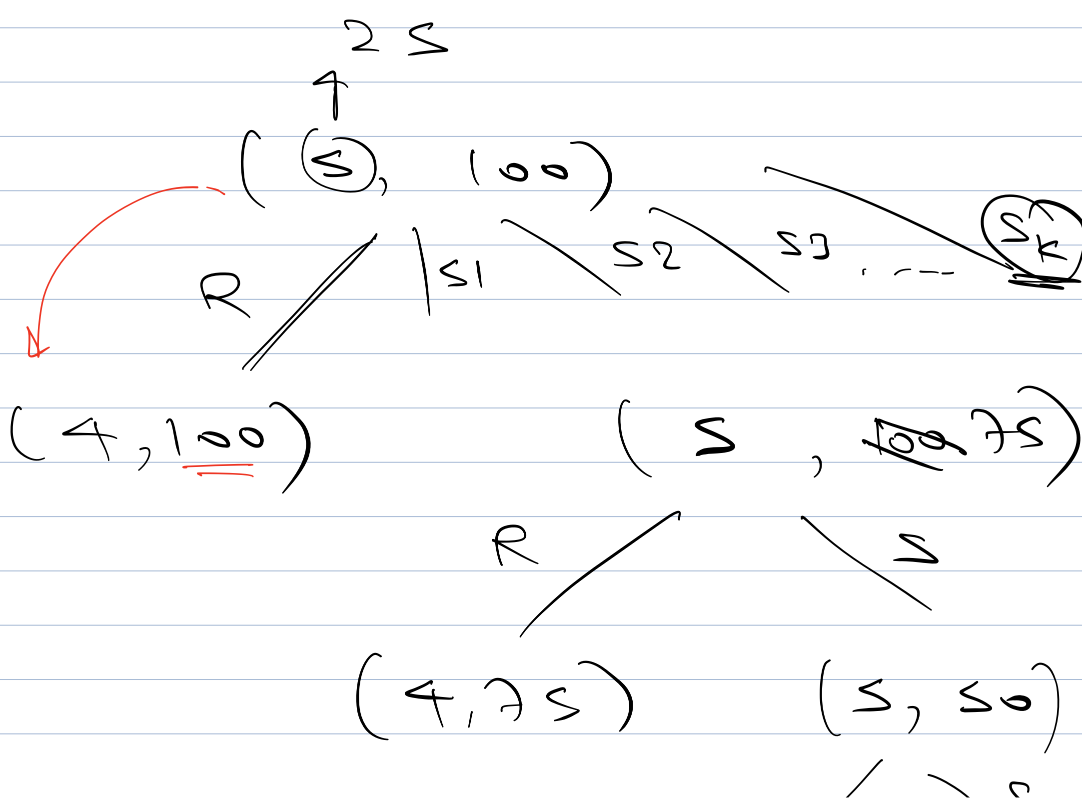
```
DP[c+1] = {0}
for (i=1; i<=c; i++) {   → c
```

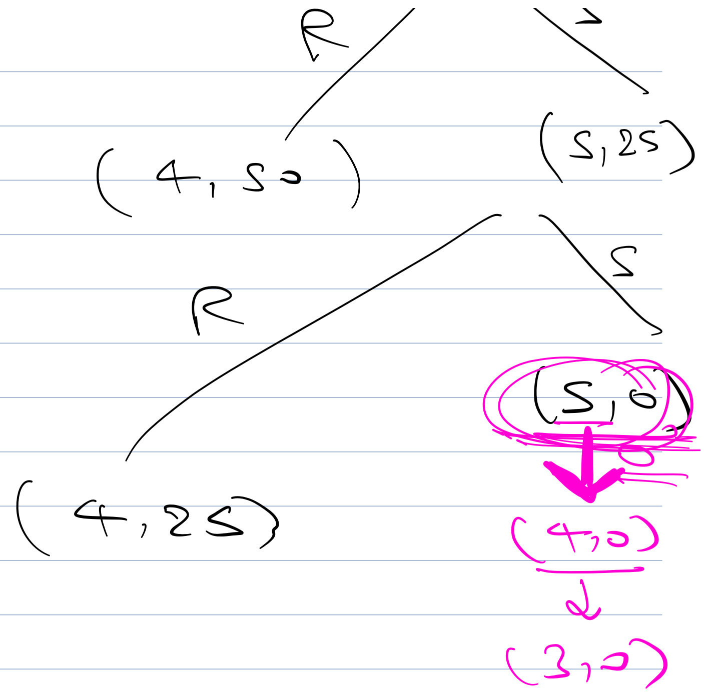
```
maxHappiness = 0;
for (j=0; j<N; j++) { → N
    if (i >= W[j]) {
        ans = H[j] + DP[i-W[j]];
        maxHappiness = max(
            ans, maxHappiness);
    }
}
DP[i] = maxHappiness;

}
```

$T.C. = O(N \times C)$

$S.C. = O(C)$





WB , MCM , LIS

PS → Ayush Sharma