Knapsack Problems → Given N objects with their values (profit/loss) "v_i" & weight "w_i". A bag is given with capacity W that can be used to carry objects st.

total sum of weights of selected objects $\leq W$ → condition

& sum of profit in the bag is maximized. → goal.

bag

1) **Fractional Knapsack** (objects can be divided) ✓

Q→ Given N cakes with their happiness (→ profit) & weight.
Find max total happiness that can be kept in a bag with capacity W.

Eg → N = 5
W = 40

	1	2	3	4	5
h →	3	8	10	2	5
w →	10	4	20	8	15

Let's consider 1 unit of weight

1 → 10 parts with h = 0.3 ✓
2 → 4 parts with h = 2 ✓✓
3 → 20 parts with h = 0.5 ✓✓
4 → 8 " " h = 0.25
5 → 15 " " h = 0.33 ✓✓

57

selects max (W=40) happiness parts.

4 * (2) + 20 * (0.5) + 15 * (0.33) + 1 * (0.3) = 8 + 10 + 5 + 0.3 = 23.3 (Ans) ✓

[4 + 20 + 15 = 39]

W = 30 (capacity of bag)

4 * (2) + 20 * (0.5) + 6 * (0.33) = 8 + 10 + 1.98 = 19.98 (Ans) ✓

Select max h[i]/w[i] cakes in descending order. ✓

Greedy ✓

TC = O(N log(N)) SC = O(1)

2) 0-1 Knapsack (objects cannot be divided)

Q→ N toys → happiness & weight.
Find max total happiness that can be kept in a bag with capacity W.

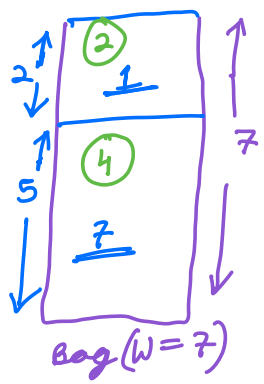
$\forall i\ w[i] > 0$

profit

Eg→ N = 4, W = 7

	1	2	3	4
h →	4	1	5	7
w →	3	2	4	5

Ans = 4+5 = 9



i	h[i]/w[i]	
1 →	4/3 = 1.33	✓ 2 X
2 →	1/2 = 0.5	✓ 4
3 →	5/4 = 1.25	✓ 3 X
4 →	7/5 = 1.4	✓ 1 choice

total happiness via greedy sol. = 7+1 = 8 ✗

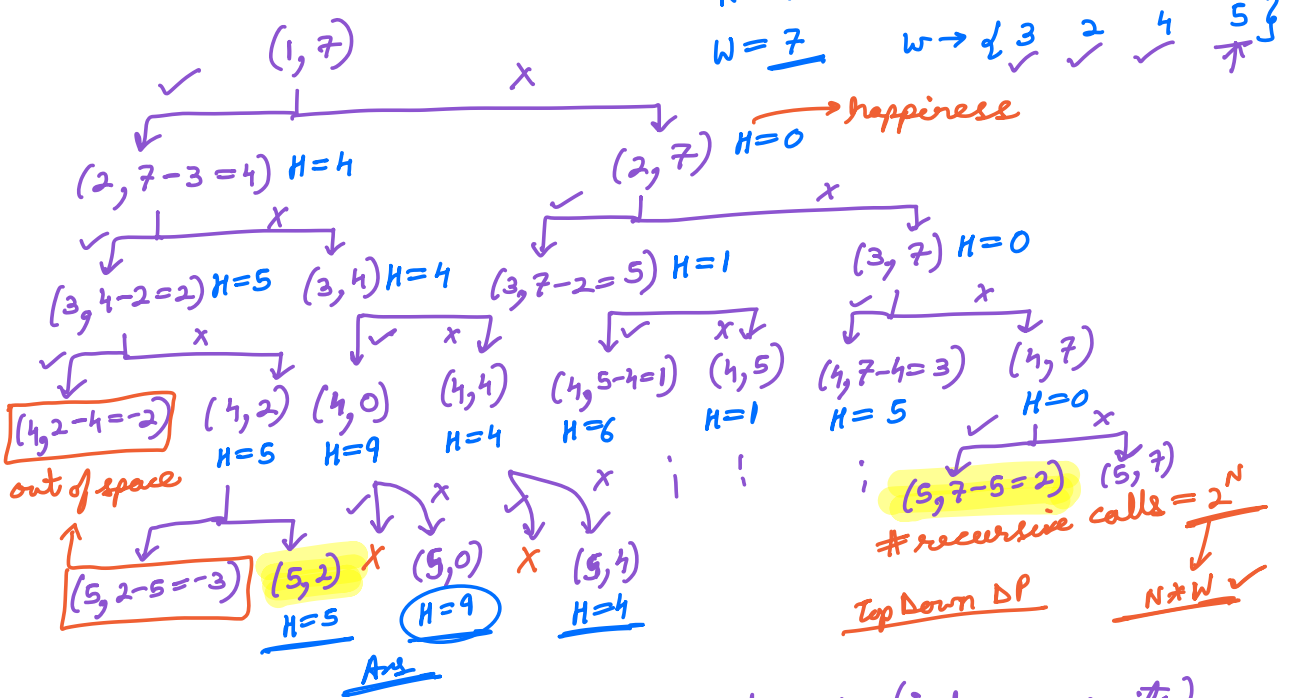
∀ toys → select / reject

Total # ways to select/reject N toys = 2^N

→ (index, capacity)

N = 4, W = 7

	1	2	3	4
h →	4	1	5	7
w →	3 ✓	2 ✓	4 ✓	5 ✗

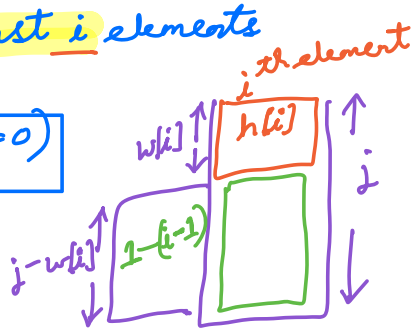


Every state can be uniquely identified with → (index, capacity)
∴ Total # unique state = N*W

index: 1 to N; capacity: 1 to W

TC = O(N*W) SC = O(N*W)

$dp[i][j]$ = max happiness considering first i elements & capacity = j.



Base condition → $dp[i][j] = 0$ if $(i == 0 \,||\, j == 0)$

Ans → $dp[N][W]$

$dp[i][j]$ → select ✓ → $h[i] + dp[i-1][j-w[i]]$

$dp[i][j]$ → reject ✗ → $dp[i-1][j]$ → ∵ i^{th} element is rejected, happiness from $(i-1)$ elements with capacity j is $dp[i][j]$.

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

```
∀i,j dp[i][j] = 0;
for (i = 1; i <= N; i++) {      // toy      ✓
    for (j = 1; j <= W; j++) {  // happiness ✓
        if (j >= w[i]) {        ✓
            dp[i][j] = max(dp[i-1][j], h[i] + dp[i-1][j-w[i]]);
        } else {
            dp[i][j] = dp[i-1][j];
        }
    }
}
return dp[N][W];
```

N = 4, W = 7

i →	1	2	3	4
h →	4	1	5	7
w →	3 ✓	2 ✓	4 ✓	5 ✓

N\W (j →)	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
✓1	0	0	0	4	4	4	4	4
i ↓ x2	0	0	1	4	4	5	5	5
✓3	0	0	1	4	5	5	6	9
x4	0	0	1	4	5	7	7	9

$dp[4][5] = 7 + dp[3][5-5]$

Ans

$i = 1 \text{ to } 3$

$j = 1 \text{ to } 7$

$dp[2][2] = 1 + dp[2-1][2-2]$

$dp[2][3] = dp[2-1][3]$

$dp[2][5] = 1 + dp[2-1][5-2]$

$dp[3][7] = 5 + dp[2][7-4]$

TC = O(N*W)
SC = O(N*W) ⇒ O(2W) = O(W) ✓

Q→ How to know which toys to select that give max happiness?

Ans → dp[N][W]

```
if(dp[N][W] == dp[N-1][W])
        Nth object is rejected;   N--;
else if(dp[N][W] == h[N] + dp[N-1][W-w[i]])
        Nth object is selected;
        N--;   W = W - w[i];
```

for i = N — 1

Ans = {1, 3} ✓

---

3) Unbounded Knapsack / 0-N Knapsack (objects can be selected multiple times)

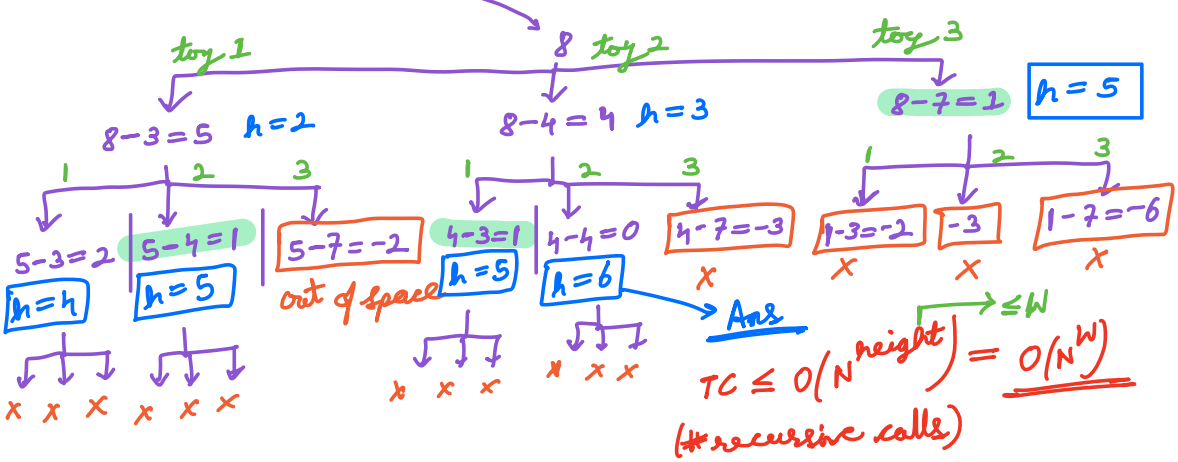
0-1 Knapsack → select ✓ → processed
→ reject ✓ → processed

0-N Knapsack → select → can select again
→ reject ✓ → processed

Q→ N = 3 toys
W = 8 capacity

	1	2	3
h →	2	3	5
w →	3	4	7

Ans = 2*3 = 6 ✓



dp[i] = max happiness that can be achieved if capacity = i

↓ capacity/weight

Base condition → dp[0] = 0

Ans = dp[W]

$$dp[i] = \max_{\forall j \text{ objects/toys}} (h[j] + dp[i - w[j]])$$

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

TC = O(W * N) SC = O(W)