Knap Sack Problems

Given N objects with their values V_i & their weights w_i.
A bag is given with capacity W that can be used to carry some objects such that →

total sum of object weights $\le$ W, and
sum of values in the bag is maximised.

① Fractional KnapSack

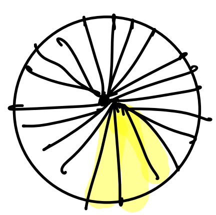
Given N cakes with their happiness and weight.
Find max total happiness that can be kept in a bag with capacity = W. (cakes can be divided)

N = 5.
W = 40

happiness[ ] → [3 8 10 2 5]
weight [ ] → [10 4 20 8 15]

$\Sigma h = 3+2+5 = 10+8 = 18+1.5 = 19.5$

$\Sigma w = 10+8+15 = 33+4 = 37 + 3 = 40$



20 → 10
1 → $\frac{10}{20}$
3 → $\frac{10}{20} * 3 = 1.5$

N = 5.
W = 40

happiness[ ] → [4 8 10 2 5]
weight [ ] → [4 4 20 8 16]

$\Sigma h = 8+10+5 = 23$

$\Sigma w = 4+20+16 = 40$

N = 5.
W = 40

happiness[] → [4 8 10 2 5] $\Sigma h = 24$ ✓
weight [] → [4 4 20 8 16] $\Sigma w = 36.$

Greedy on only happiness doesn't work.

N = 5.
W = 40

happiness[] → [3 8 10 2 5]
weight [] → [10 4 20 8 15]

10 pastries, $h = 0.3$
4 pastries, $h = 2$
20 pastries, $h = 0.5$
8 pastries, $h = 0.25$
15 pastries, $h = 0.33$

pick 40 pastries.

idea. → Select the cakes in descending order w.r.t $\frac{\text{happiness}}{\text{weight}}$.
(Greedy)

① Sort cakes on basis of $\frac{\text{happiness}}{\text{wt}}$.

② Keep on selecting the cakes if current wt. $\leq$ capacity of bag.
(fractional cake can also be there)

$\left[\begin{array}{l} T.C \to O(N \log N) \\ S.C \to O(1) \end{array}\right]$

0-1 KnapSack. (object can't be divided) ✓ ✗

Q) Given N toys with their happiness & weight. find max total happiness that can be kept in a bag with capacity = W. (toys can't be divided)

$N = 4$

$W = 7$

$h \to [\ 4 \quad 1 \quad 5 \quad 7\ ]$

$wt \to [\ 3 \quad 2 \quad 4 \quad 5\ ]$

$h/wt \to \quad 1.33 \quad 0.5 \quad 1.25 \quad 1.4$

$\left\{ \begin{array}{l} \Sigma h = 8 \\ \Sigma w = 7. \end{array} \right.$ Greedy on just happiness. ✗

{ Greedy on h/wt is also not working }

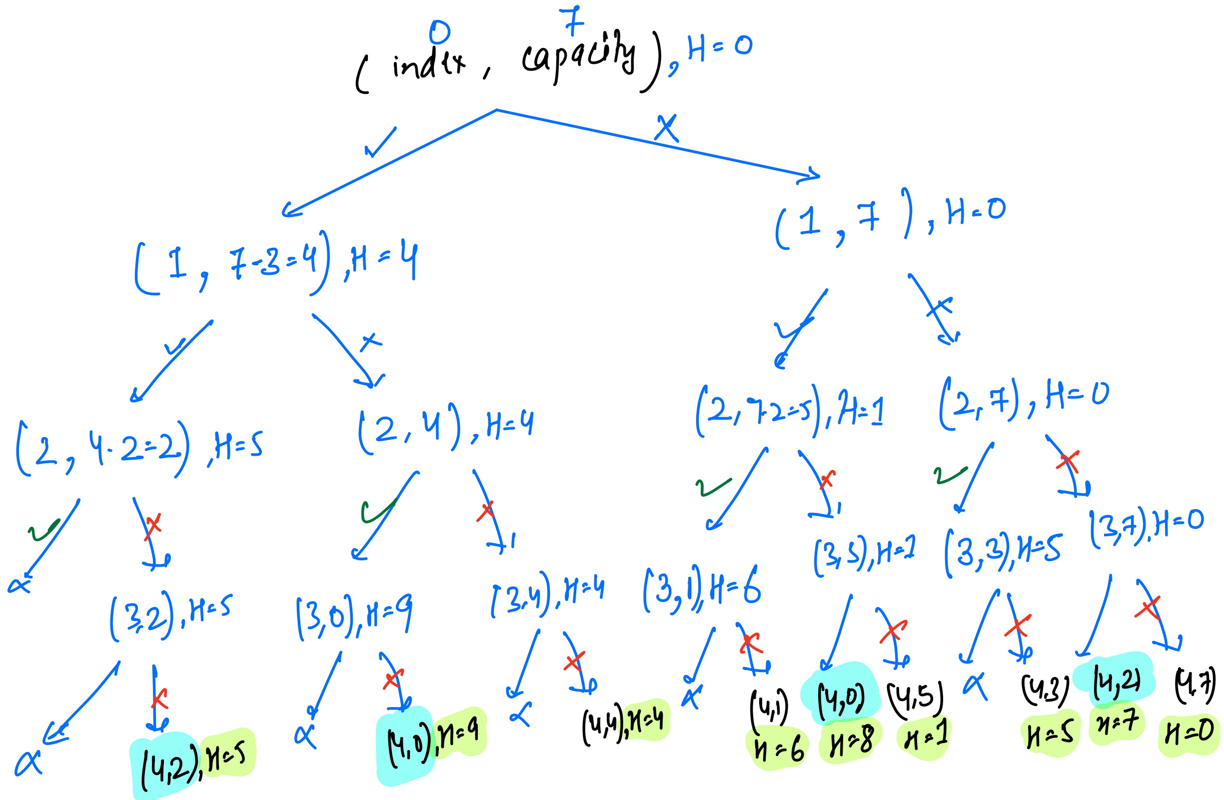
Brute-force ⇒ Consider all the subsets with $\Sigma wt \le W$ and select the one with max happiness.

$\{ T.C \to O(2^N) \}$

N = 4

W = 7

$$
\begin{array}{lcccc}
 & 0 & 1 & 2 & 3 \\
h \rightarrow & [\,4 & 1 & 5 & 7\,] \\
wt \rightarrow & [\,3 & 2 & 4 & 5\,]
\end{array}
$$



optimal sub-structure ✓

overlapping sub-problems ✓ } D.P.

no. of toys

$$\text{maxVal}(N, W)$$

✓ → $\text{happiness}_n + \text{maxVal}(N-1, W - w_n)$

× → $\text{maxVal}(N-1, W)$

$$\text{MaxValue}(N, W) = \underline{\max}\begin{cases} 0 + \text{maxVal}(N-1, W), \\ \text{happiness}_n + \text{maxVal}(N-1, W - w_n) \end{cases}$$

```
int dp [N+1] [W+1] ;
```

Base cases → $N = 0$ ⇒ no toys ⇒ 0 happiness

$W = 0$ ⇒ no space ⇒ 0 happiness.

Top-down

no. of toys → i, capacity → j

```
int maxValue( int i, int j , int[][] dp){
        if (i == 0 || j == 0) { return 0}
        if(dp[i][j] != -1) {return dp[i][j] }

        // pick the element.
        if( wt[i-1] <= j){
            // if weight of ith toy is less than cap.
            dp[i][j] = max [ 0 + maxVal( i-1, j) ,
                             h[i-1] + maxVal(i-1, j-wt[i-1]) ]
        } else {
            dp[i][j] = max( i-1, j) ;
        }
        return dp[i][j];
}
```

{ T.C → $O(N*W)$
S.C → $O(N*W)$ }

{ dp[i][j] → max happiness with i toys and j capacity. }

Bottom-up:

W = 8.

N = 5.

h[ ] : 12 20 15 6 10

wt[ ] : 3 6 5 2 4

0 1 2 3 4

Capacity.

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0	0	12	12	12	12	12	12
2	0	0	0	12	12	12	20	20	20
3	0	0	0	12	12	15	20	20	27
4	0								
5	0								

no. of toys

h	wt	index
12	3	0
20	6	1
15	5	2
6	2	3
10	4	4

12, 15 + 0

dp[i-1][j-5]

dp[2][0]

ans.

dp[i][j] → Max happiness with first i toys and j capacity.

$$dp[i][j] = \begin{cases} 0 + dp[i-1][j] \\ h[i-1] + dp[i-1][j - wt[i-1]] \\ h[2] + dp[2][8-5] \end{cases}$$

$i = 3$, $j = 8$

pseudo-code.

```
int dp[N+1][W+1]
// initalise 0th row & 0th col with 0.
for( i = 1 ; i <= N ; i++) {
        for( j = 1 ; j <= W ; j++) {
                dp[i][j] = dp[i-1][j];
                if ( wt(i-1) <= j ) {
                        dp[i][j] = max( dp[i-1][j],
                                        h[i-1] + dp[i-1][j - wt[i-1]] )
                }
        }
}
return dp[N][W];
```

T.C => $O(N*W)$
S.C => $O(N*W)$

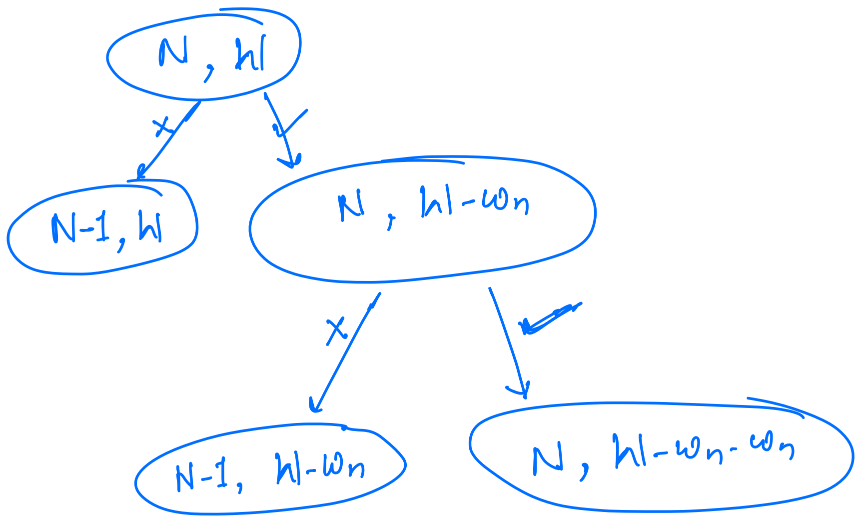
Un-bounded Knapsack

(You can pick any element any no. of times)

value :	2	3	5
wt :	3	4	7

$W = 8$, $N = 3$.

ans = 6.



$$\boxed{maxValue(N, W) = \max\begin{pmatrix} 0 + maxVal(N-1, W), \\ val[i-1] + maxVal(N, W - wt[i-1]) \end{pmatrix}}$$

Q. Given an array with +ve elements. Flip sign of some of its elements such that sum of elements of final array is minimum non-negative integer. Find min elements to flip.

A = [10 15 6 3 3]
-15 -3

[ans = flip 2 elements.]

$A[i] \to$ flip
$A[i] \to$ don't flip

[0-1 Knapsack]

sum = S $\Rightarrow$ $\boxed{\sum \text{flipped elements} <= S/2}$

W

value[i] → 1.

wt[i] → A[i].

A = [10 15 6 3 3] Sum = 37.

$\sum$ flipped elements $\leq \frac{37}{2} = 18.$

H.W.