Knapsack Problem

Given N objects with their values Vi profit/loss their weight Wi. A bag is given with capacity W that can be used to carry some objects such that the total sum of object weights W and sum of profit in the bag is maximized or sum of loss in the bag is minimized.

We will try Knapsack when these combinations are given:

- number of objects will be N ✓
- every object will have 2 attributes namingly value and weight ✓
- and capacity will be given ✓

Fractional Knapsack (objects can be divided)

Q → N cakes → happiness, → weight

Find max total happiness that can be kept in a bag with capacity W. (cakes can be divided)

$$\begin{array}{lllllll} & & \downarrow & \checkmark & \checkmark & & \checkmark \\ & & 0 & 1 & 2 & 3 & 4 \\ N = 5 & h = [& 3 & 8 & 10 & 2 & 5\,] \\ W = 40 & w = [& 10 & 4 & 20 & 8 & 15\,] \\ & & \underline{0.3} & & & & \end{array}$$

Only focus h[i] →

H = 10

= 18

= 23

$\underline{23.3}$

W = 40 − 20 = 20

20 − 4 = 16

16 − 15 = 1

N = 3

W = 10

h = [50 30 30]

w = [10 5 5]

10 * 5   5 * 6 ✓

5 * 6 ✓

H = $\underline{60}$ (Ans)

Sol → Select items in descending order of h[i]/w[i].

Greedy

$TC = O(N \log(N))$ $SC = O(N)$

0-1 Knapsack (division of objects is not allowed)

Q → N toys → happiness
→ weight

Find max total happiness that can be kept in a bag with capacity W.

✓ ✓
1 2 3 4

$N = 4$ $h = [4 \quad 1 \quad 5 \quad 7]$ $H = 7 + 1 = 8$ ✗

$W = 7$ $w = [3 \quad 2 \quad 4 \quad 5]$ $W = 7 - 5 = 2 - 2 = 0$

$Ans = 4 + 5 = \underline{9}$

✗ ✓ ✓

$N = 3$ $h = [50 \quad 30 \quad 30]$ $Ans = \underline{60}$

$W = 10$ $w = [\textcircled{6} \quad 5 \quad 5]$

$h/w = 8.\_ \quad 6 \quad 6$

$W = 10 - 6 = \underline{4}$

Bruteforce → consider all subsets.

$TC = \underline{O(2^N)}$

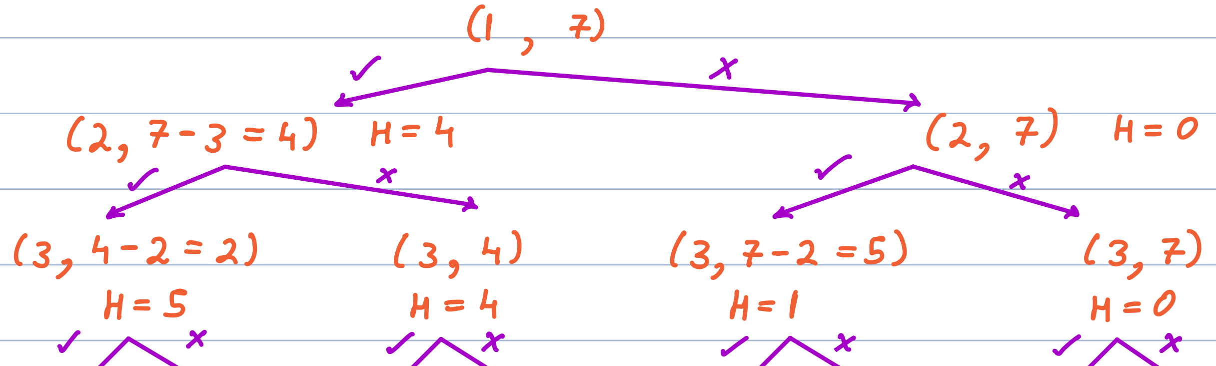
↓
1 2 3 4

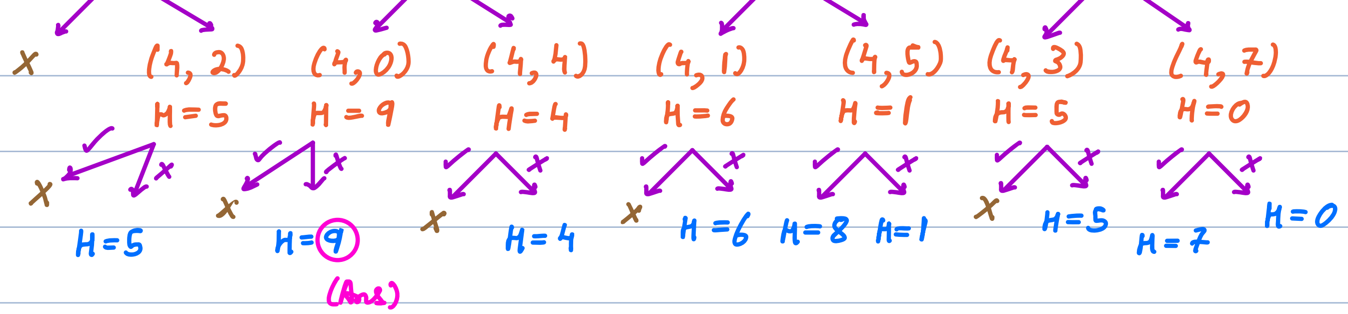
$N = 4$ $h = [4 \quad 1 \quad 5 \quad 7]$

$W = 7$ $w = [3 \quad 2 \quad 4 \quad \textcircled{5}]$

→

(index, capacity) H = 0





0 — (N-1)

0 — W

Total unique states possible = $\underline{N*(W+1)}$ → $\underline{O(N*W)}$

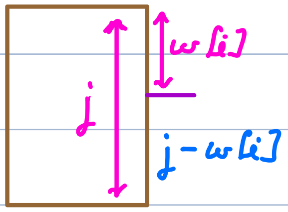
optimal substructure ✓

overlapping subproblems ✓ } $\underline{DP}$

dp[i][j] → max total happiness considering first 'i' items & capacity 'j'.

dp[i][j] $\xrightarrow{\times}$ dp[i-1][j]

✓ → h[i] + dp[i-1][j-w[i]]



dp[0][j] = 0 // no object

dp[i][0] = 0 // no capacity

```
for i → 0 to N {
    for j → 0 to W {
        if (i==0 || j==0)   dp[i][j] = 0
        else if (j >= w[i]) {
            dp[i][j] = max (dp[i-1][j], dp[i-1][j - w[i]] + h[i])
        } else {
            dp[i][j] = dp[i-1][j]   // reject i^th item
        }
    }
```

```
} return dp[N][W]
```

TC = $\underline{O(N * W)}$

SC = $\underline{O(N * W)} \rightarrow \underline{O(2 * W)}$

only store 2 rows

Unbounded / 0 – N Knapsack

(objects cannot be divided)

(object can be selected multiple times)

Q → N toys → happiness, weight

Find max total happiness that can be kept in a bag with capacity W. Infinite supply of every toy is available.

✓✓

N = 3

W = 8

$h = \begin{array}{ccc} 1 & 2 & 3 \\ [2 & 3 & 5] \end{array}$

$w = [3 \quad 4 \quad 7]$

Ans = 6

N = 2

W = 100

$h = \begin{array}{cc} 1 & 2 \\ [1 & 30] \end{array}$

$w = [1 \quad 50]$

100 times

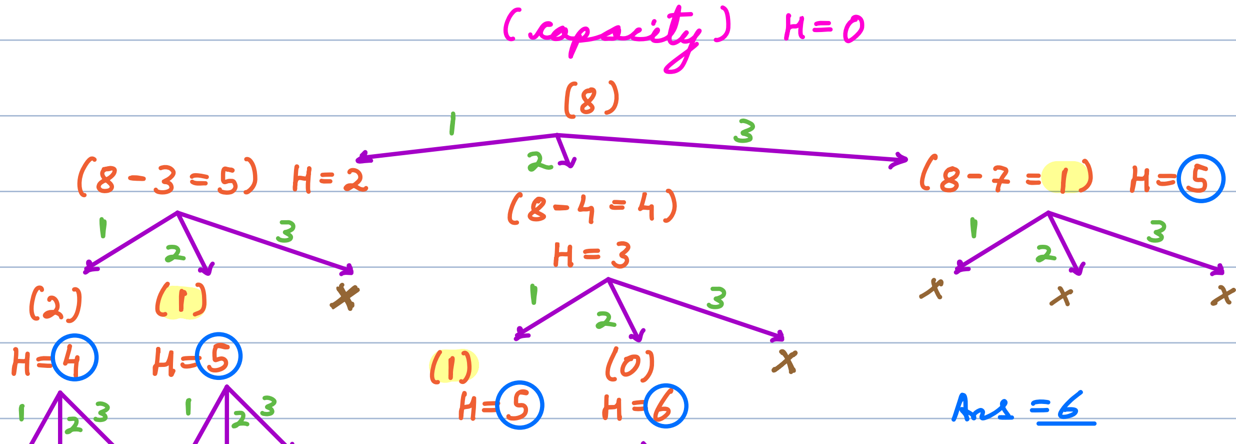
Ans = 100

N = 3

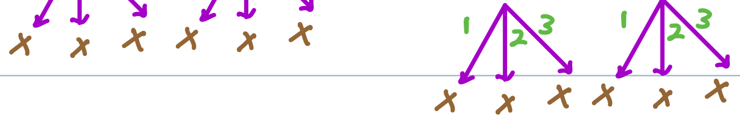
W = 8

$h = \begin{array}{ccc} 1 & 2 & 3 \\ [2 & 3 & 5] \end{array}$

$w = [3 \quad 4 \quad 7]$



Ans = 6



Total unique states = W+1

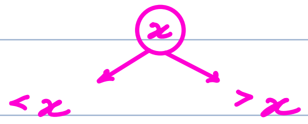
optimal substructure ✓
overlapping subproblems ✓ } DP

dp[i] → max total happiness with capacity i
dp[0] = 0

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

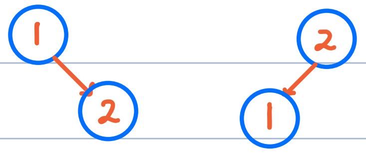
TC = O(N * W)
SC = O(W)

Q → Given N distinct nodes, find total # unique BST that can be formed.

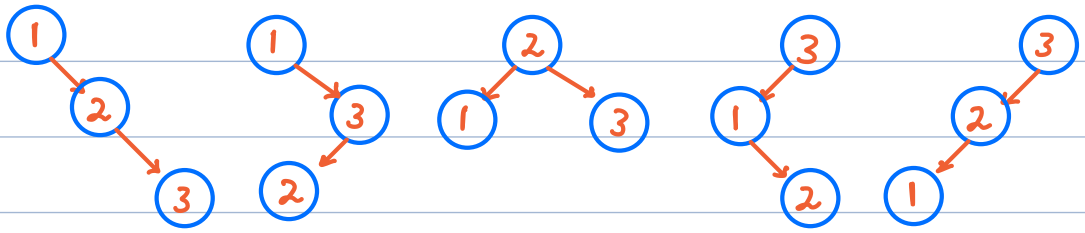


N	Ans
1	1
2	2
3	5

{1, 2}

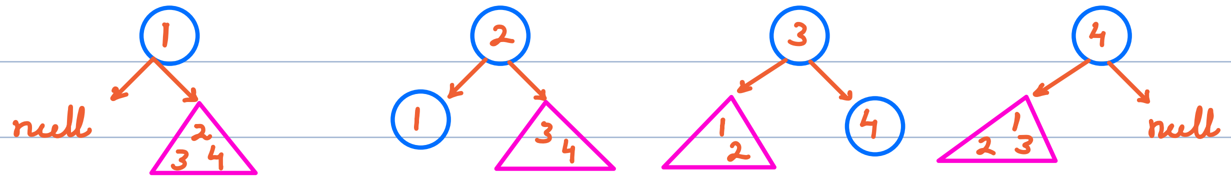


{1, 2, 3}



All nodes can be root node.

$N = 4$



$$c(4) = c(0) * c(3) + c(1) * c(2) + c(2) * c(1) + c(3) * c(0)$$

$$1 * 5 + 1 * 2 + 2 * 1 + 5 * 1 = \underline{14}$$

$$\boxed{c(N) = \sum_{i=0}^{N-1} c(i) * c(N-i-1)} \rightarrow \underline{\text{Catalan Number}}$$

$$\frac{{}^{2N}C_N}{(N+1)} = \frac{2N!}{N!\ (N+1)!}$$ ✓

```
c[0] = 1      c[1] = 1
for i → 2 to N {
    for j → 0 to (i-1) {
        c[i] += c[j] * c[i-j-1]
    }
} return c[N]
```

$TC = \underline{O(N^2)}$ ✓ $SC = \underline{O(N)}$