Today's content

-> Rod cutting

-> Coin change

-> 0-1 Knapsack

Q1. Given a rod of length N and an array of length N.

price[i] → price of i-length rod.

Find the max price that can be obtained by cutting the rod into 1 or more pieces and selling them.

length	Price
5	6
4 + 1	6
3 + 2	6
3 + 1 + 1	4
2 + 2 + 1	9
2 + 1 + 1 + 1	7
1 + 1 + 1 + 1 + 1	5

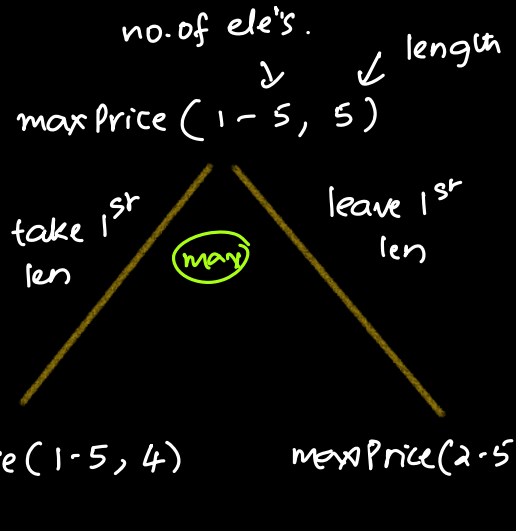
N = 5.

price : [1 4 2 5 6]

length: 1 2 3 4 5

Idea: Greedy won't work.

Idea:



Unbounded Knapsack.

Q2. Coin change

Given N- different denominations

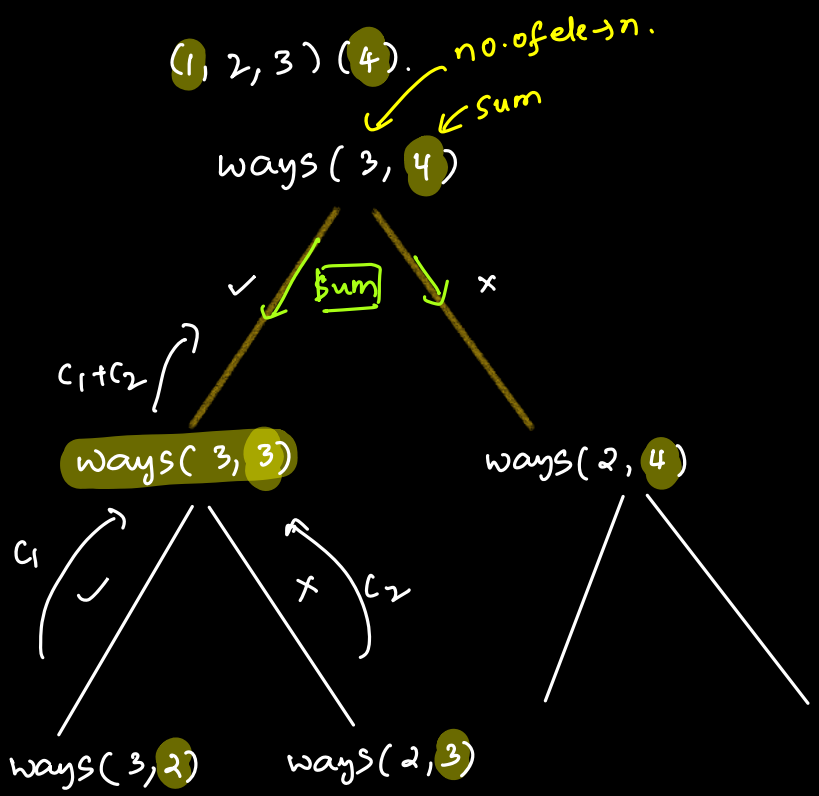
Find total no. of ways to pay a given amount

Note: Single denomination can be used more than once.

Amount = 4, denominations [1 2 3]

[1 1 1 1]
[1 1 2]
[1 3]
[2 2]

4 ways = ans



Same as unbounded Knapsack, But instead of max, take sum of two subproblems.

Base case:

```
if (s==0)
    return 1

if (s<0)
    return 0

if (n==0)
    return 0
```

0-1 Knapsack V2 [NP - hard problems]

Q3. Given N toys, with their happiness & weight.

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

Note: toys cannot be divided.

Ideas we've discussed.

1. using recursion, generating all possibilities.

   TC: $O(2^N)$; 2^{10}.

2. using DP.

   TC: $O(N * W)$ { not a polynomial soln }.

   $N \downarrow 10$; $W = 10^9$.

   TC: $10 * 10^9$.

NP hard problems. [not deterministic polynomial].

It's extremely diff. to find a common algo which works efficiently for all sorts of i/p.

Idea 3: Another method. [Let's say you've the below data]

max^m happiness I can get with W = 9 =>

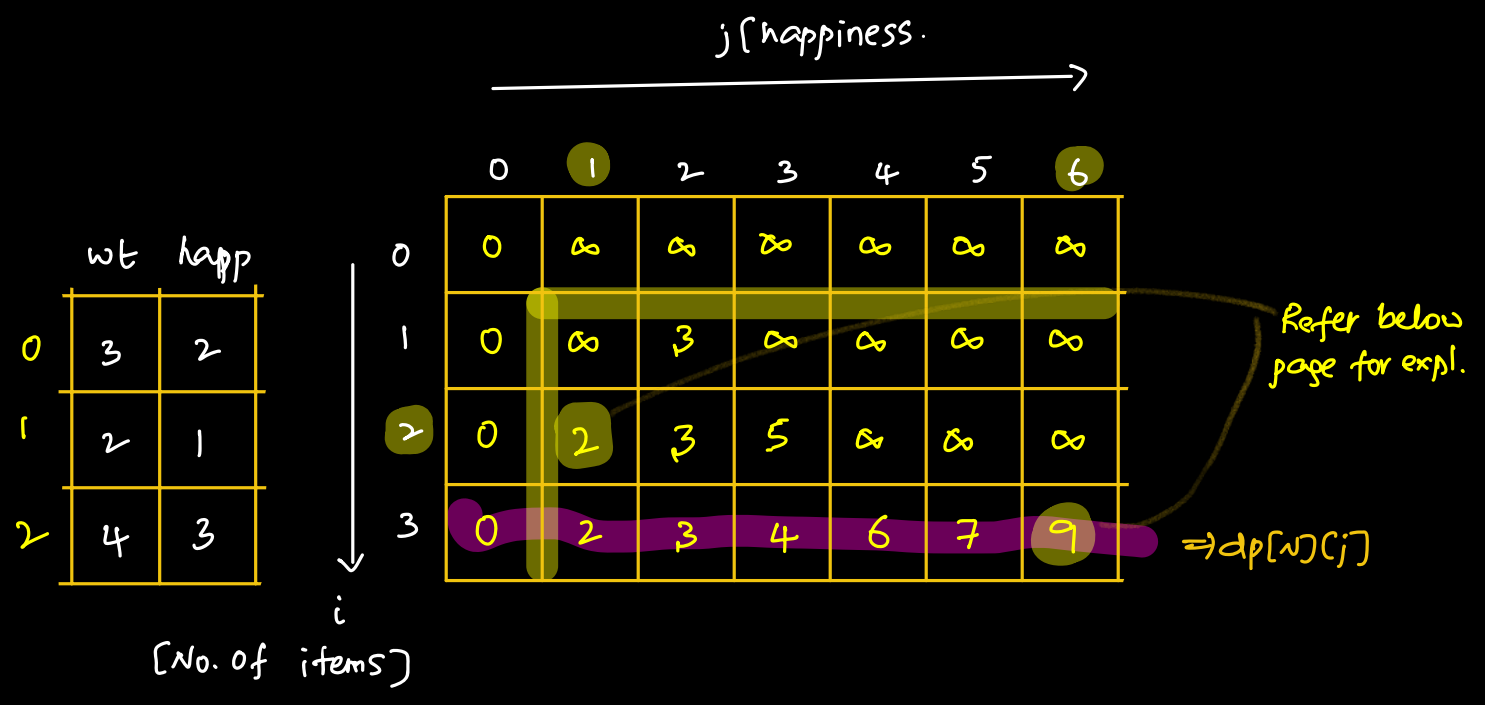
dp[i][j] → Minimum wt required to get a happiness of 'j' using first 'i' elements.

happ : [2 1 3]
wt : [3 2 4]

w = 7.

max happiness (ignoring weights) : [2 + 1 + 3] = 6 . (Σ happ[i])

dp → size[4][7].



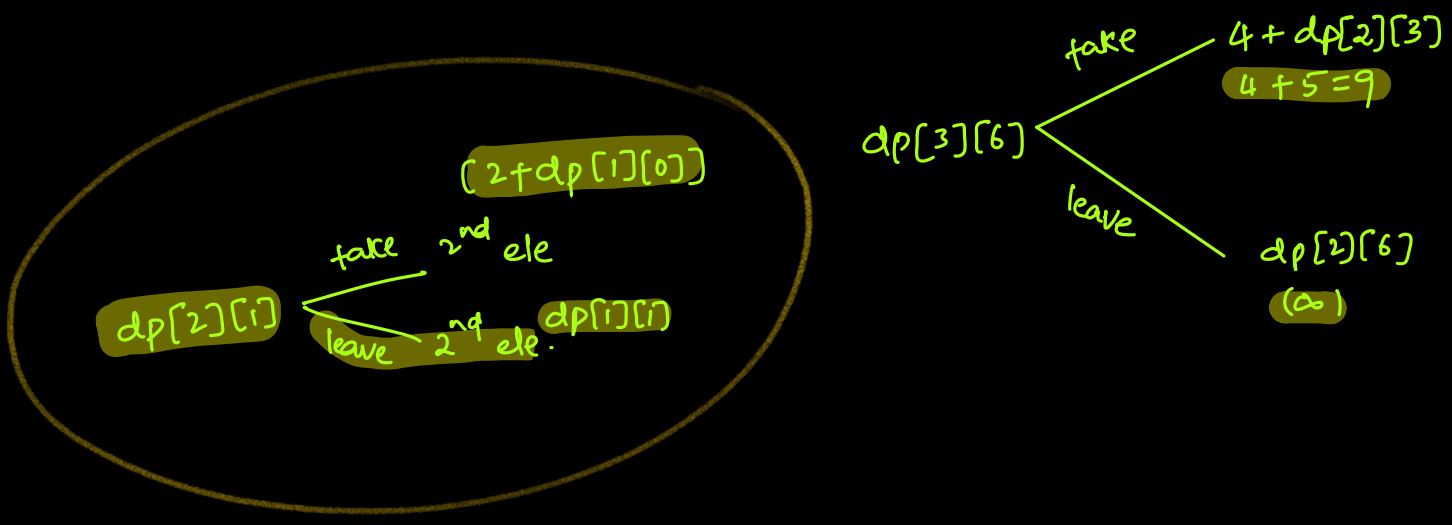
dp[i][j] = min { don't select the i^{th} ele ; f1 ⇒ dp[i-1][j]
Select i^{th} ele ; f2 ⇒ wt[i-1] + dp[i-1][j - happ[i-1]] }

first 3 elements

happiness : 0 1 2 3 4 5 6
minWtRequired : 0 2 3 4 6 7 9
✓ ✗

w = 7.

⇒ ans = 5.



Code

```
Sum = 0
for (i=0; i<N; i++)
    Sum += happ[i]

dp[N+1][sum+1]
// fill first row → ∞   (0 index)
// fill first col → 0

for(i=1 ; i<=N; i++)
    for ( j=1; j<= sum ;j++)
        dp[i][j]= dp[i-1][j] // f1
        if ( j >= happ[i-1] && dp[i-1][j-happ[i-1]] != ∞ )
            dp[i][j] = min { dp[i][j],
                             wt[i-1] + dp[i-1][j-happ[i-1]] }
                                       f2

int ans=0
for(j= sum ; j >=0 ;j--)
    if (dp[N][j] <= w)
        ans=j ; break
```

return ans;

Constraints

$1 \le N \le 500$

$1 \le W \le 10^9$

$1 \le wt[i] \le 10^9$

$1 \le happ[i] \le 50$

TC: O(N * sum)

SC: O(N * sum)

TC: $500 * (500 * 50)$

: $125 * 10^6$

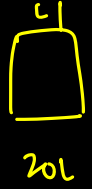
= $1.25 * 10^8$.

TC: $O(2^N)$ => $2^{500} \ge 10^8$. X

$O(N * W)$ => $500 * 10^9 \ge 10^8$ X

Car. 20L .

Type1 Tell me budget.



Type2 Don't tell me budget.

L1 < L2 < L3 < L4