coin change :- n different denominations.

(K) ⇒ no of ways to get a change of k

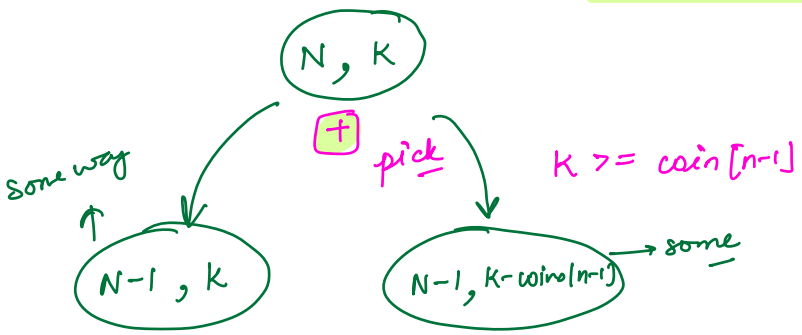
can't include a coin more than once

7 4 9 6 10 13 11 14

k = 22

7 4 11
13 9
7 9 6

Q. n, integers-array
count no of ways in which subset sum = k



dp[i][j] = no of ways to achieve j change with i available coins

$$dp[i][j] = \begin{cases} dp[i-1][j] & \\ + & \\ dp[i-1][j-coin[i-1]] & , \ j >= coin[i-1] \end{cases}$$

[0-0] ↑ i

Base case

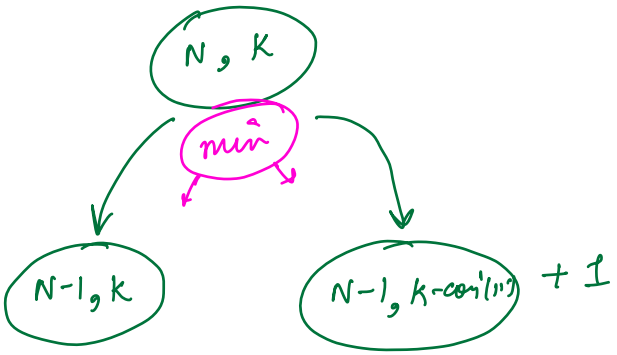
$\forall_{j=1}^{j<=K} \ dp[0][j] = 0$

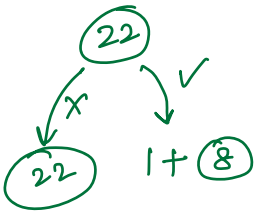
$\forall_{i=0}^{i<=n} \ dp[i][0] = 1$ → required change

dp[0][0] = 1

min no of coins required for change of k

Q min no of elements to pick such that subset sum = k





Base case:

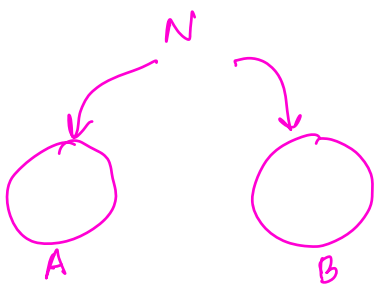
$\forall\, i <= n \quad dp[i][0] = 0$

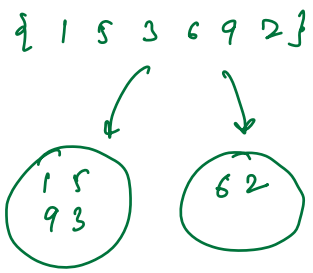
$\forall\, j <= k \quad dp[0][j] = \infty$ (INT_MAX) → n+1

acc to your const

Q array - N integers. Divide all elements into two subsets.

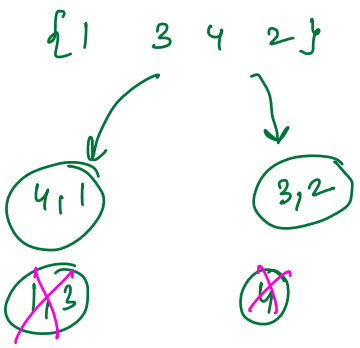
(i) Find no of ways in which you can divide such that both subsets have equal sum.





Sum(A) = sum(B)

Every element must belong to one of sets



sum(A) = sum(B)

sum(A) + sum(B) = totalsum

sum(A) + sum(A) = totalsum

sum(A) = totalsum / 2

totalsum is even

(ii) divide diff of the sum of sets = k

True / false
no of ways

sum(A) - sum(B) = k

sum(A) + sum(B) = totalsum

+

2 sum(A) = k + totalsum

sum(A) = (k + totalsum) / 2

(iii) minimise the abs diff of two sets

{ 1, 5, 7 } { 3 2 4 7 6 3 }

1,5 7 7,6 3,2,4,3

1

dp[N+1][sum/2+1]

totalsum = 100

diff

1	50	51	dp[N][50]
3	49	52	dp[N][49]

diff	S1	S2
0	50	50
2	49	51
4	48	52

dp[N][50] → T / F

dp[N][49] → T / F

dp[N][48]

dp[N][sum/2] → dp[N][sum/2 - 1]

prepare dp table using knapsack

dp[N+1][sum/2+1]

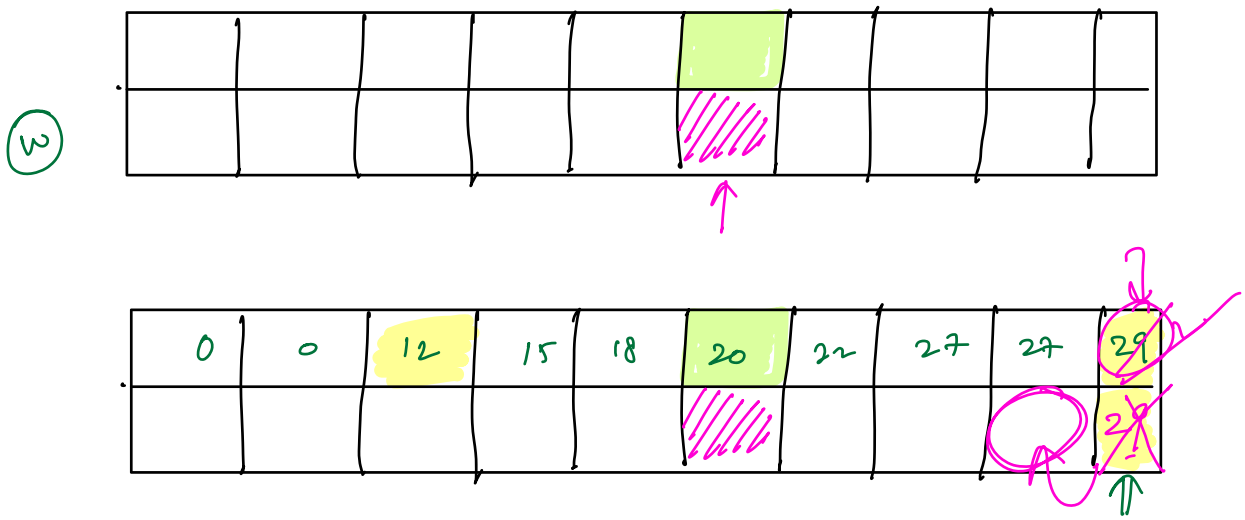
```
K = sum/2

while ( K >=0)
{
        if ( dp[N][K] == true)
        {
                s1 = K;
                s2 = sum - K;
                return abs(s1-s2);
        }
        K--;
}
```

Knapsack

S.C: O(n*w) — 2D dp table

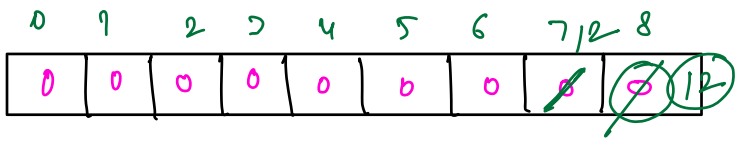
n*w → 2*w → w

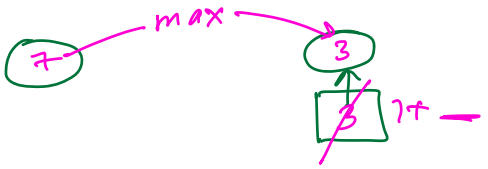
dp(i,j) = (i-1, j) (i-1, j-w)



w		0	1	2	3	4	5	6	7	8
	0	0	0	0	0	0	0	0	0	0
12	3	1	0	0	0	12	12	12	12	12	12
20	6	2	0	0	0	12	12	12	20	20	20
15	5	3	0	0	0	12	12	15	20	20	15+12=27
6	2	4	0	0	6	12	12	18	20	21	27
10	4	5	0	0	6	12	12	18	20	22	27

for a particular cell, same row data is not req





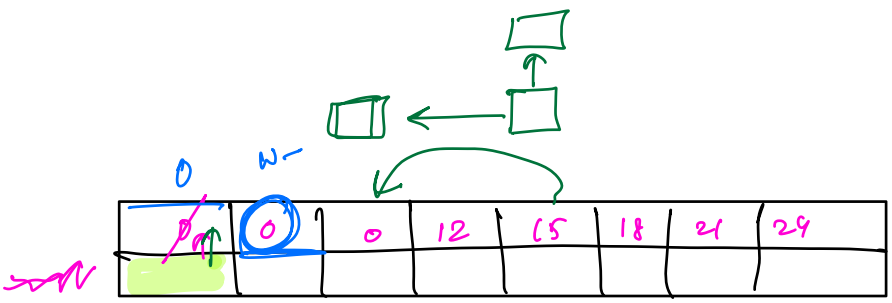
dp[w+1]

i-1, j

```
for (int i=0; i<=n; i++)
{
    for(j=w; j>=0; j--)
    {
        if( j>= weight(i-1])
        {
            dp[j]= max( dp[j],
                value[i-1] + dp[j-weight[i-1]])
        }
    }
}
```

[0-∞] Knapsack dp[i][j] dp[i-1][j]

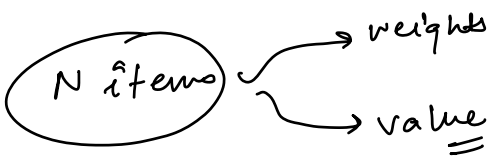
dp[i][j-...]

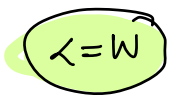


```
for (int i=0; i<=n; i++)
{
    for(j=0 ; j<=W ; j++)        // l to R
    {
        if ( j >= weight(i-1))
        {
            dp[j] = max( dp[j],
                value[i] + dp[j-weight[i-1]])
        }
    }
}
```

[0~∞]

#





$1 \leq N \leq 500$

$1 \leq value[i] \leq 50$

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

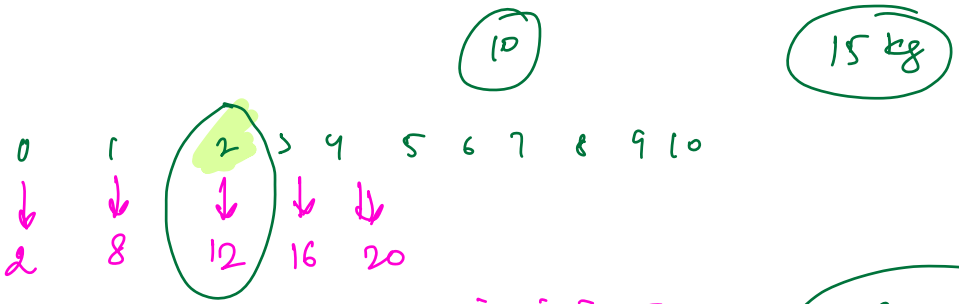
$1 \leq W \leq 10^9$

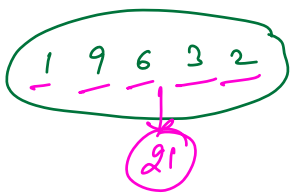
max value

N, W — max value

N, value — min weight

max possible value = 500 * 50 = 25000

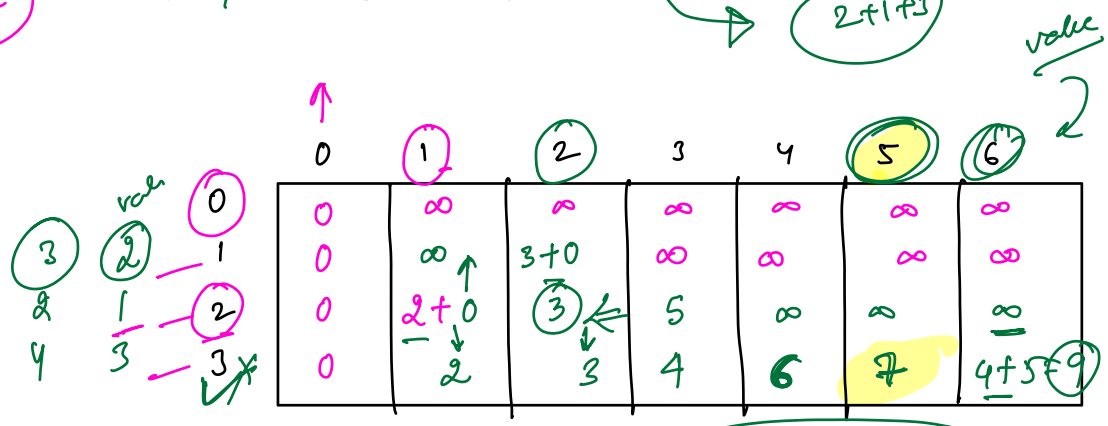




N = 3
W = 7

value = 2 1 3
weights = 3 2 4

0 → 6
6 ← 2+1+j



N = 7

dp[N][j] <= W

ans = max j

dp[n+1][max value + 1]

sum of all values

N * max value[i]

dp[i][j] = min(dp[i-1][j], weight[i-1] + dp[i-1][j - value[i-1]])