0/1 knapsack problem

↳ Given a set of items, each with a weight and a value, determine the maximum value that can be obtained by selecting a subset of items, subject to a total weight constraint. (i.e. the total weight of selected items should not exceed the capacity of the knapsack)

eg:

	W	Value / Profit	P/W
I1	1	6	$6/1 = 6$
I2	2	10	$10/2 = 5$
I3	3	12	$12/3 = 4$

Capacity = 5 units

Fractional knapsack :- Max Profit $= 6 \times 1 + 5 \times 2 + 4 \times 2$

$= 6 + 10 + 8$

$= \boxed{24}$

0/1 knapsack :- I1 + I2, I1 + I3, I2 + I3

16, 18, 22

max Profit = $\boxed{22}$

$$\text{maxProfitknapsack}(n, W) = \begin{cases} 0, & n = 0 \text{ or } W = 0 \\ mPK(n-1, W), & w[n] > W \\ \max\begin{pmatrix} mPK(n-1, W), \\ v[n] + mPK(n-1, W-w[n]) \end{pmatrix}, & w[n] \le W \end{cases}$$

DP approach:-

```
                                                        [n]            [n]
                                                        weight         value of
                            # items   capacity          of items       items
function  maxProfitKnapsack ( n,        W,              weights,       values ):

    dp = matrix of size (n+1) x (W+1) with first row        <- dp[i][j]:- The max
         and first column initialized to 0                     profit if you have
                                                               first 'i' items &
                                                               capacity of 'j'
    for (i: 1 -> n):
        for (j: 1 -> W):
            if weights[i-1] <= j :
                dp[i][j] = max( dp[i-1][j],
              i^th item
                             values[i-1] + dp[i-1][j - weights[i-1]] )
                             i^th item     i-1 items   i^th item
            else :
                dp[i][j] = dp[i-1][j]

    return dp[n][W]
```

eg:

	w		values	
I1	2	w[0]	3	v[0]
I2	3	w[1]	4	v[1]
I3	4	w[2]	5	v[2]
I4	5	w[3]	6	v[3]

Knapsack capacity (W) = 5

Sol.

v_{i-1}	w_{i-1}		item ↓ / dp / cap →	0	1	2	3	4	5
			0	0	0	0	0	0	0
3	2	I1	1	0	0	3	3	3	3
4	3	I2	2	0	0	3	4	4	7
5	4	I3	3	0	0	3	4	5	7
6	5	I4	4	0	0	3	4	5	$\boxed{7}$

Max Profit = 7

Items taken:- I_1, I_2

dp[3][5] w[2] = 4
v[2] = 5

7 dp[2][5]   5 + dp[2][1]

i j
dp[1][4]

one items, capacity = 4
first items

I: w = 5, v = 6 | II: w = 3, v = 5

I w > j:
dp[0][4]

II w ≤ j:
max (dp[0][4] , 5 + dp[0][1])

dp[3][4] 4
3rd items

Time :- O(n·W)
items, capacity

Space :- O(n·W)

eg:

Consider the following set of items, each with a given weight and value:

- Item 1: weight = 1, value = 1
- Item 2: weight = 3, value = 4 }
- Item 3: weight = 4, value = 5 }
- Item 4: weight = 5, value = 7

The maximum weight capacity of the knapsack is W = 7.

W= 7

14, 8

23, 9

Select all that apply:

A. The maximum value that can be achieved with the given capacity is 9. ✓

B. If you include Item 4, you cannot achieve the maximum value. ✓

C. The optimal solution includes Item 2 and Item 3. ✓

D. If you exclude Item 1, the maximum value you can achieve is 9. ✓

E. The optimal solution includes Item 1, Item 3, and Item 4. ✗

[GATE CS 2018]

Consider the weights and values of items listed below. Note that there is only one unit of each item.

Item number	Weight (in Kgs)	Value (in rupees)	v/w
1	10	60	60/10 = 6
2	7	28	28/7 = 4
3	4	20	20/4 = 5
4	2	24	24/2 = 12

The task is to pick a subset of these items such that their total weight is no more than 11 Kgs and their total value is maximized. Moreover, no item may be split. The total value of items picked by an optimal algorithm is denoted by V_{opt}. A greedy algorithm sorts the items by their value-to-weight ratios in descending order and packs them greedily, starting from the first item in the ordered list. The total value of items picked by the greedy algorithm is denoted by V_{greedy}.

The value of $V_{opt} - V_{greedy}$ is ____

$60 - 44 = \boxed{16}$

V_{opt} :

$1 \to \boxed{60}$ 2,3 48 2,4 52 3,4 44

④ $V_{opt} = 60$

$\cancel{1}$ ⑤ $V_{greedy} = 24 + 20 = 44$

Subset Sum Problem

↳ Given a set of non-negative integers and a value Sum, determine if there is a subset with sum equal to the given sum.

eg. $S = \{3, 34, 4, 12, 5, 2\}$ Sum $= 9$

o/p :- True $\{4, 5\}$ or $\{3, 4, 2\}$

Brute Force :- check all possible subsets, $O(2^n)$

DP approach:-

```
function isSubsetSum (set, n, sum):
    dp = boolean matrix of size (n+1)x (sum+1)
    for (i: 0 → n):          // First Column
        dp [i][0] = True
    for (i: 1 → sum):        // First Row
        dp[0][i] = False

    for (i: 1 → n):
        for (j: 1 → sum):
            if set [i-1] > j :  → leave
                dp [i][j] = dp [i-1][j]
            if set [i-1] ≤ j :  → take it or leave it
                dp[i][j] = dp[i-1][j] OR dp[i-1][j- set[i-1]]
                             leave                take
    return dp[n][sum]
```

dp[5][0]

$T = O(n \cdot sum)$

$S = O(n \cdot sum)$

↳ can be reduced to $O(\min(n, sum))$

eg: $S = \{3, 34, 4, 12, 5, 2\}$ Sum = 9

ele ↓ (i)	dp (j: Sum →)	0	1	2	3	4	5	6	7	8	9
	0	T	F	F	F	F	F	F	F	F	F
3	1	T	F	F	T	F	F	F	F	F	F
34	2	T	F	F	T	F	F	F	F	F	F
4	3	T	F	F	T	T	F	F	T	F	F
12	4	T	F	F	T	T	F	F	T	F	F
5	5	T	F	F	T	T	T	F	T	T	T
2	6	T	F	T	T	T	T	T	T	T	T

$\{5, 4\}$

$\{2, 4, 3\}$

[GATE CS 2008]

The subset-sum problem is defined as follows. Given a set of n positive integers, $S = \{a_1, a_2, a_3, \ldots, a_n\}$, and positive integer W, is there a subset of S whose elements sum to W? A dynamic program for solving this problem uses a 2-dimensional Boolean array, X, with n rows and $W + 1$ columns. $X[i, j], 1 \leq i \leq n, 0 \leq j \leq W$, is TRUE, if and only if there is a subset of $\{a_1, a_2, \ldots, a_i\}$ whose elements sum to j.

Which of the following is valid for $2 \leq i \leq n$, and $a_i \leq j \leq W$?

A. $X[i, j] = X[i-1, j] \vee X[i, j-a_i]$
B. $X[i, j] = X[i-1, j] \vee X[i-1, j-a_i]$
C. $X[i, j] = X[i-1, j] \wedge X[i, j-a_i]$
D. $X[i, j] = X[i-1, j] \wedge X[i-1, j-a_i]$

[GATE CS 2008]

The subset-sum problem is defined as follows. Given a set of n positive integers, $S = \{a_1, a_2, a_3, \ldots, a_n\}$, and positive integer W, is there a subset of S whose elements sum to W? A dynamic program for solving this problem uses a 2-dimensional Boolean array, X, with n rows and $W + 1$ columns. $X[i, j], 1 \leq i \leq n, 0 \leq j \leq W$, is TRUE, if and only if there is a subset of $\{a_1, a_2, \ldots, a_i\}$ whose elements sum to j.

Which entry of the array X, if TRUE, implies that there is a subset whose elements sum to W?

A. $X[1, W]$
B. $X[n, 0]$
C. $X[n, W]$
D. $X[n-1, n]$