COIN CHANGE PROBLEM

DYNAMIC PROGRAMMING

Coin change problem

- Given a set of coins **v**[] for example {1,2,3}
- Given an **M** amount → the total
- How many ways the coins **v**[] can be combined in order to get the total **M**?
- The order of coins does not matter !!!
- This is the coin change problem

Coin change problem

Coins **v[]** → {1, 2, 3}

Total amount **M** → 4

Solution to the coin change problem:
{1,1,1,1} {1,1,2} {1,3} {2,2}

The order of coins does not matter !!!
For exmple {1,3} = {3,1}

<u>Recursion</u>

- The naive approach is to use a simple recursive method / function
- For every single coin we have two options: include it in our solution or exclude it
- <u>Problems</u>: time complexity + overlapping subproblems
- Exponential time complexity: **O(2^{N})** where **N** is the number of coins
- For every coin we have **2** options whether to take it or not

Dynamic programming

We have to create a solution matrix:

dpTable[numOfCoins+1][totalAmount+1]
rows **columns**

We have to define the base cases:

- if totalAmount is **0** → there is **1** way to make the change
  Because we do not include any coin !!!

- if numOfCoins is **0** → there is **0** way to change the amount
  In this case there is no solution !!!

Complexity: **O(v*M)**

Dynamic programming

We have to create a solution matrix:

dpTable[numOfCoins+1][totalAmount+1]
rows **columns**

For every coin: make a decision whether to include it or not

Check if the coin value is less than or equal to the amount needed

If yes → then we will find ways by including that coin and excluding that coin

1.) include the coin: reduce the amount by coin value and use the subproblem solution // **totalAmount – v[i]**

2.) exclude the coin: solution for the same amount without considering that coin

$$\mathbf{dpTable[i][j]} = \begin{cases} \mathbf{0 \text{ if } i = 0} \\ \mathbf{1 \text{ if } j = 0} \\ \mathbf{dpTable[i-1][j] + dpTable[i]\big[j - v[i-1]\big] \text{ if } v[i] \le j} \\ \mathbf{dpTable[i-1][j] \text{ if } v[i] > j} \end{cases}$$

If the coin value is smaller than the amount: it means we can consider that coin !!!

Base cases

$$\mathbf{dpTable[i][j]} = \begin{cases} \mathbf{0 \text{ if } i = 0} \\ \mathbf{1 \text{ if } j = 0} \\ \mathbf{dpTable[i-1][j] + dpTable[i]\big[j - v[i-1]\big] \text{ if } v[i] \leq j} \\ \mathbf{dpTable[i-1][j] \text{ if } v[i] > j} \end{cases}$$

How many ways the first **i** coins can be combined in order to get the total **j**?

If the coin value is greater than the amount: it means we can not consider that coin !!!

M = 4
v[] = {1,2,3}

numOfColumns = M+1
numOfRows = v.length+1

	0	**1**	**2**	**3**	**4**
0					
1					
2					
3					

totals

coins

M = 4
v[] = {1,2,3}

numOfColumns = M+1
numOfRows = v.length+1

	0	**1**	**2**	**3**	**4**
0					
1					
2					
3					

totals

coins

<u>Subproblems</u>: we consider the totals {0,1,2,3,4} step by step when we can have {0,1,2,3} coins at the same time !!! We solve the subproblems and combine them for the final solution

M = 4
v[] = {1,2,3}

numOfColumns = M+1
numOfRows = v.length+1

		M=0	M=1	M=2	M=3	M=4	totals
Take no coins	0						
Take 1	1						
Take 1 or 2	2						
Take 1 or 2 or 3	3						
	coins						

Subproblems: we consider the totals {0,1,2,3,4} step by step when we can have {0,1,2,3} coins at the same time !!! We solve the subproblems and combine them for the final solution

M = 4
v[] = {1,2,3}

numOfColumns = M+1
numOfRows = v.length+1

	0	**1**	**2**	**3**	**4**
0	0	0	0	0	0
1					
2					
3					

M = 4
v[] = {1,2,3}

numOfColumns = M+1
numOfRows = v.length+1

	0	1	2	3	4
0	1	0	0	0	0
1	1				
2	1				
3	1				

M = 4
v[] = {1,2,3}

numOfColumns = M+1
numOfRows = v.length+1

	0	**1**	**2**	**3**	**4**
0	1	0	0	0	0
1	1	1			
2	1				
3	1				

M = 4
v[] = {1,2,3}

numOfColumns = M+1
numOfRows = v.length+1

	0	**1**	**2**	**3**	**4**
0	1	0	0	0	0
1	1	1	1		
2	1				
3	1				

M = 4
v[] = {1,2,3}

numOfColumns = M+1
numOfRows = v.length+1

	0	**1**	**2**	**3**	**4**
0	1	0	0	0	0
1	1	1	1	1	
2	1				
3	1				

M = 4
v[] = {1,2,3}

numOfColumns = M+1
numOfRows = v.length+1

	0	**1**	**2**	**3**	**4**
0	1	0	0	0	0
1	1	1	1	1	1
2	1				
3	1				

M = 4
v[] = {1,2,3}

numOfColumns = M+1
numOfRows = v.length+1

	0	**1**	**2**	**3**	**4**
0	1	0	0	0	0
1	1	1	1	1	1
2	1				
3	1				

dpTable[i][j] = $\begin{cases} \mathbf{dpTable}[\mathbf{i}-\mathbf{1}][\mathbf{j}] + \mathbf{dpTable}[\mathbf{i}][\mathbf{j}-\mathbf{v}[\mathbf{i}-\mathbf{1}]] \text{ if } \mathbf{v}[\mathbf{i}] \leq \mathbf{j} \\ \mathbf{dpTable}[\mathbf{i}-\mathbf{1}][\mathbf{j}] \text{ if } \mathbf{v}[\mathbf{i}] > \mathbf{j} \end{cases}$

What does it mean simply?
If the given **v[i] > j** → copy the content of the box above the current
Else: **dpTable[i][j] = value of box above the current**
+ (value in same row – v[i])

M = 4
v[] = {1,2,3}

numOfColumns = M+1
numOfRows = v.length+1

	0	**1**	**2**	**3**	**4**
0	1	0	0	0	0
1	1	1	1	1	1
2	1	1			
3	1				

dpTable[i][j] = $$\begin{cases} \mathbf{dpTable[i-1][j] + dpTable[i][j - v[i-1]] \text{ if } v[i] \leq j} \\ \mathbf{dpTable[i-1][j] \text{ if } v[i] > j} \end{cases}$$

M = 4
v[] = {1,2,3}

numOfColumns = M+1
numOfRows = v.length+1

	0	**1**	**2**	**3**	**4**
0	1	0	0	0	0
1	1	1	1	1	1
2	1	1	2		
3	1				

dpTable[i][j] = $\begin{cases} \mathbf{dpTable[i-1][j] + dpTable[i]\left[j - v[i-1]\right] \text{ if } v[i] \leq j} \\ \mathbf{dpTable[i-1][j] \text{ if } v[i] > j} \end{cases}$

dpTable[2][2] = dpTable[1][2] + dpTable[2][0]

M = 4
v[] = {1,2,3}

numOfColumns = M+1
numOfRows = v.length+1

	0	1	2	3	4
0	1	0	0	0	0
1	1	1	1	1	1
2	1	1	2	2	
3	1				

$$\textbf{dpTable[i][j] =} \begin{cases} \mathbf{dpTable[i-1][j] + dpTable[i]\big[j - v[i-1]\big]\ if\ v[i] \leq j} \\ \mathbf{dpTable[i-1][j]\ if\ v[i] > j} \end{cases}$$

dpTable[2][3] = dpTable[1][3] + dpTable[2][1]

M = 4
v[] = {1,2,3}

numOfColumns = M+1
numOfRows = v.length+1

	0	**1**	**2**	**3**	**4**
0	1	0	0	0	0
1	1	1	1	1	1
2	1	1	2	2	3
3	1				

dpTable[i][j] = $\begin{cases} \mathbf{dpTable[i-1][j] + dpTable[i][j-v[i-1]]\ if\ v[i] \le j} \\ \mathbf{dpTable[i-1][j]\ if\ v[i] > j} \end{cases}$

dpTable[2][4] = dpTable[1][4] + dpTable[2][2]

M = 4
v[] = {1,2,3}

numOfColumns = M+1
numOfRows = v.length+1

	0	**1**	**2**	**3**	**4**
0	1	0	0	0	0
1	1	1	1	1	1
2	1	1	2	2	3
3	1	1			

dpTable[i][j] = $$\begin{cases} \mathbf{dpTable[i-1][j] + dpTable[i]\left[j - v[i-1]\right]\ if\ v[i] \leq j} \\ \mathbf{dpTable[i-1][j]\ if\ v[i] > j} \end{cases}$$

dpTable[3][1] = dpTable[2][1]

M = 4
v[] = {1,2,3}

numOfColumns = M+1
numOfRows = v.length+1

	0	**1**	**2**	**3**	**4**
0	1	0	0	0	0
1	1	1	1	1	1
2	1	1	2	2	3
3	1	1	2		

dpTable[i][j] = $\begin{cases} \mathbf{dpTable[i-1][j] + dpTable[i]\big[j - v[i-1]\big]\ if\ v[i] \leq j} \\ \mathbf{dpTable[i-1][j]\ if\ v[i] > j} \end{cases}$

dpTable[3][2] = dpTable[2][2]

M = 4
v[] = {1,2,3}

numOfColumns = M+1
numOfRows = v.length+1

	0	**1**	**2**	**3**	**4**
0	1	0	0	0	0
1	1	1	1	1	1
2	1	1	2	2	3
3	1	1	2	3	

dpTable[i][j] = $$\begin{cases} \mathbf{dpTable[i-1][j] + dpTable[i]\big[j - v[i-1]\big]\ if\ v[i] \leq j} \\ \mathbf{dpTable[i-1][j]\ if\ v[i] > j} \end{cases}$$

dpTable[3][3] = dpTable[2][3] + dpTable[3][0]

M = 4
v[] = {1,2,3}

numOfColumns = M+1
numOfRows = v.length+1

	0	**1**	**2**	**3**	**4**
0	1	0	0	0	0
1	1	1	1	1	1
2	1	1	2	2	3
3	1	1	2	3	4

dpTable[i][j] = $$\begin{cases} \mathbf{dpTable[i-1][j] + dpTable[i]\left[j - v[i-1]\right] \text{ if } v[i] \leq j} \\ \mathbf{dpTable[i-1][j] \text{ if } v[i] > j} \end{cases}$$

dpTable[3][4] = dpTable[2][4] + dpTable[3][1]

M = 4
v[] = {1,2,3}

numOfColumns = M+1
numOfRows = v.length+1

	0	**1**	**2**	**3**	**4**
0	1	0	0	0	0
1	1	1	1	1	1
2	1	1	2	2	3
3	1	1	2	3	4

$$\textbf{dpTable[i][j]} = \begin{cases} \mathbf{dpTable}[\mathbf{i}-\mathbf{1}][\mathbf{j}] + \mathbf{dpTable}[\mathbf{i}]\left[\mathbf{j}-\mathbf{v}[\mathbf{i}-\mathbf{1}]\right] \text{ if } \mathbf{v}[\mathbf{i}] \leq \mathbf{j} \\ \mathbf{dpTable}[\mathbf{i}-\mathbf{1}][\mathbf{j}] \text{ if } \mathbf{v}[\mathbf{i}] > \mathbf{j} \end{cases}$$

SO WE HAVE SOLVED OUR PROBLEM WITHOUT RECALCULATING THE SAME PROBLEMS OVER AND OVER AGAIN !!!