GREEDY ALGORITHM TO FIND MIN NO. OF COINS

Min no. of coins Problem

Given a value V, we want to make change for V Rs. We have infinite supply of each of the denominations in Indian currency.

Min no. of coins Problem

Given a value V, we want to make change for V Rs. We have infinite supply of each of the denominations in Indian currency.

i.e., we have infinite supply of { 1, 2, 5, 10, 20, 50, 100, 500, 1000} valued coins/notes, what is the minimum number of coins and/or notes needed to make the change for V Rs?

Examples

Input: Value = 70

Output: 2

We need a 50 Rs note and a 20 Rs note.

Min no. of coins Problem

Given a value V, we want to make change for V Rs. We have infinite supply of each of the denominations in Indian currency.

i.e., we have infinite supply of { 1, 2, 5, 10, 20, 50, 100, 500, 1000} valued coins/notes, what is the minimum number of coins and/or notes needed to make the change for V Rs?

Examples

Input: Value = 70
Output: 2
We need a 50 Rs note and a 20 Rs note.

Input: Value = 121
Output: 3
We need a 100 Rs note, a 20 Rs note and a 1 Rs coin.

Solution

Greedy Approach

1) Initialize result as empty.
2) Find the largest denomination that is smaller than V.
3) Add found denomination to result.Subtract value of found denomination from V.
4) If V becomes 0, then print result.
   Else repeat steps 2 and 3 for new value of V

Illustration

Input: Value = 70

Result set: empty

Illustration

Input: Value = 70

Result set: empty

Check denominations: 1, 2, 5, 10, 20,50,100

100 is greater than 70.

Illustration

Input: Value = 70

Result set: empty

Check denominations: 1, 2, 5, 10, 20,50,100

100 is greater than 70.

So, push 50 in the result set

New Value= 70-50 = 20

Illustration

Value = 20

Result set: {50}

Check denominations: 1, 2, 5, 10, 20

We found 20 which is equal to Value

So, push 20 in the result set

New Value= 20-20 = 0

Illustration

Value = 0

Result set: {50,20}

Since Value is 0 now, we'll print the result

Illustration

Input: Value = 70

Output: 2

We need a 50 Rs note and a 20 Rs note.

Implementation

```
// Driver program
void findMin(int V)
{
    // Initialize result
    vector<int> ans;

    // Traverse through all denomination
    for (int i=n-1; i>=0; i--)
    {
        // Find denominations
        while (V >= deno[i])
        {
            V -= deno[i];
            ans.push_back(deno[i]);
        }
    }

    // Print result
    for (int i = 0; i < ans.size(); i++)
            cout << ans[i] << "  ";
}
```

Time Complexity

```
// Traverse through all denomination
for (int i=n-1; i>=0; i--)
{
    // Find denominations
    while (V >= deno[i])
    {
        V -= deno[i];
        ans.push_back(deno[i]);
    }
}
```

Worst case:
Denominations array: [1]

Time complexity: $O(V)$

V: value

CAUTION

This approach may not work for all denominations

For example, it doesn’t work for denominations {9, 6, 5, 1} and V = 11. The above approach would print 9, 1 and 1. But we can use 2 denominations 5 and 6.

For general input, we use dynamic programming approach

(1) Greedy Method

Make change for **$2.80** given the following denominations:
$1.00, $0.50, $0.20, $0.10

$2.80

$1.00	$0.50	$0.20	$0.10

(1) Greedy Method

Make change for **$2.80** given the following denominations:
$1.00, $0.50, $0.20, $0.10

$1.80

$1.00	$0.50	$0.20	$0.10

(1) Greedy Method

Make change for **$2.80** given the following denominations:

$1.00, $0.50, $0.20, $0.10

$0.80

$1.00	$0.50	$0.20	$0.10

(1) Greedy Method

Make change for **$2.80** given the following denominations:
$1.00, $0.50, $0.20, $0.10

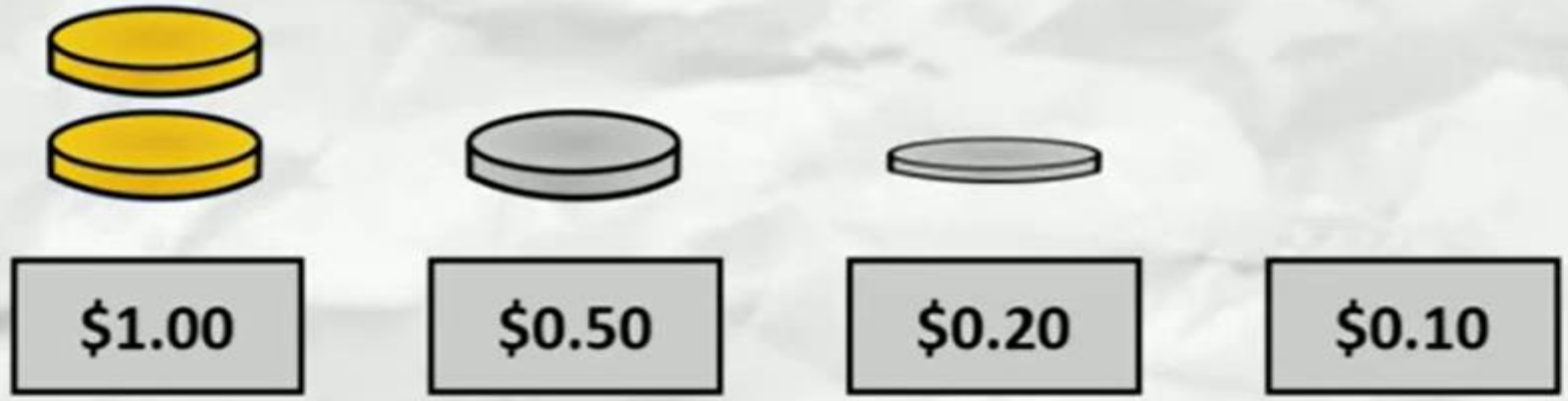
$0.30

$1.00	$0.50	$0.20	$0.10

(1) Greedy Method

Make change for **$2.80** given the following denominations:
$1.00, $0.50, $0.20, $0.10

$0.10

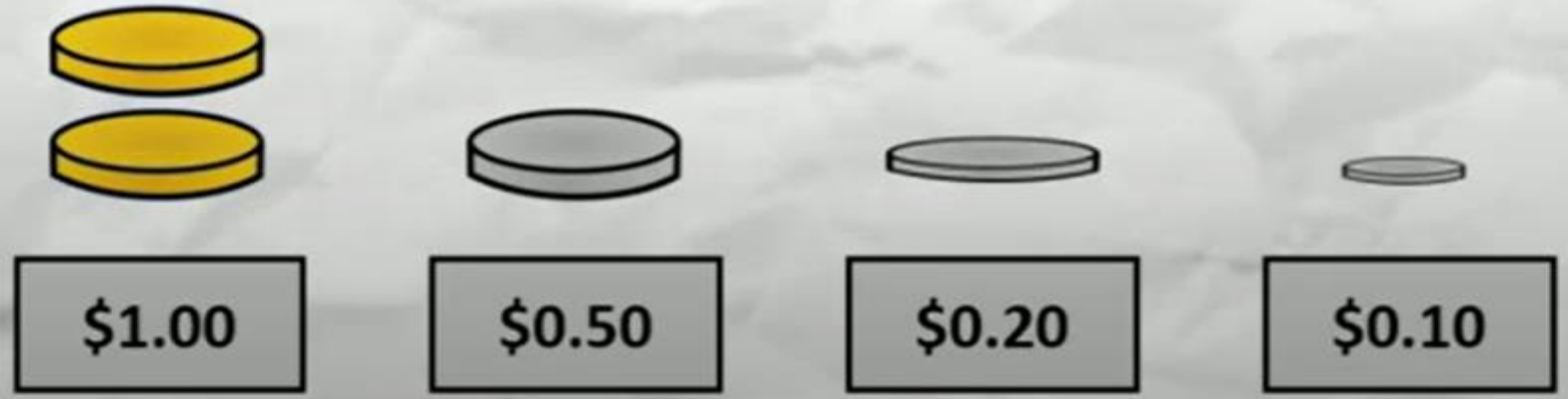


(1) Greedy Method

Make change for **$2.80** given the following denominations:

$1.00, $0.50, $0.20, $0.10

$0.00



```python
value = 280
denominations = [100,50,20,10]

taken = [0,0,0,0]    # Coins chosen
current = 0          # Current coin

while value > 0:
    if value >= denominations[current]:
        value -= denominations[current]
        taken[current] += 1
        print(value, taken)

    else:
        current += 1
```

```
180 [1, 0, 0, 0]
80 [2, 0, 0, 0]
30 [2, 1, 0, 0]
10 [2, 1, 1, 0]
0 [2, 1, 1, 1]
```

(2) Greedy Method Fails

Make change for **$0.90** given the following denominations:
$0.50, $0.30, $0.20, $0.01

```
1  value = 90
2  denominations = [50,30,20,1]
```

```
40 [1, 0, 0, 0]
10 [1, 1, 0, 0]
9 [1, 1, 0, 1]
8 [1, 1, 0, 2]
7 [1, 1, 0, 3]
6 [1, 1, 0, 4]
5 [1, 1, 0, 5]
4 [1, 1, 0, 6]
3 [1, 1, 0, 7]
2 [1, 1, 0, 8]
1 [1, 1, 0, 9]
0 [1, 1, 0, 10]
```

$0.90

$0.50

$0.30

$0.20

$0.01

$0.40

$0.50

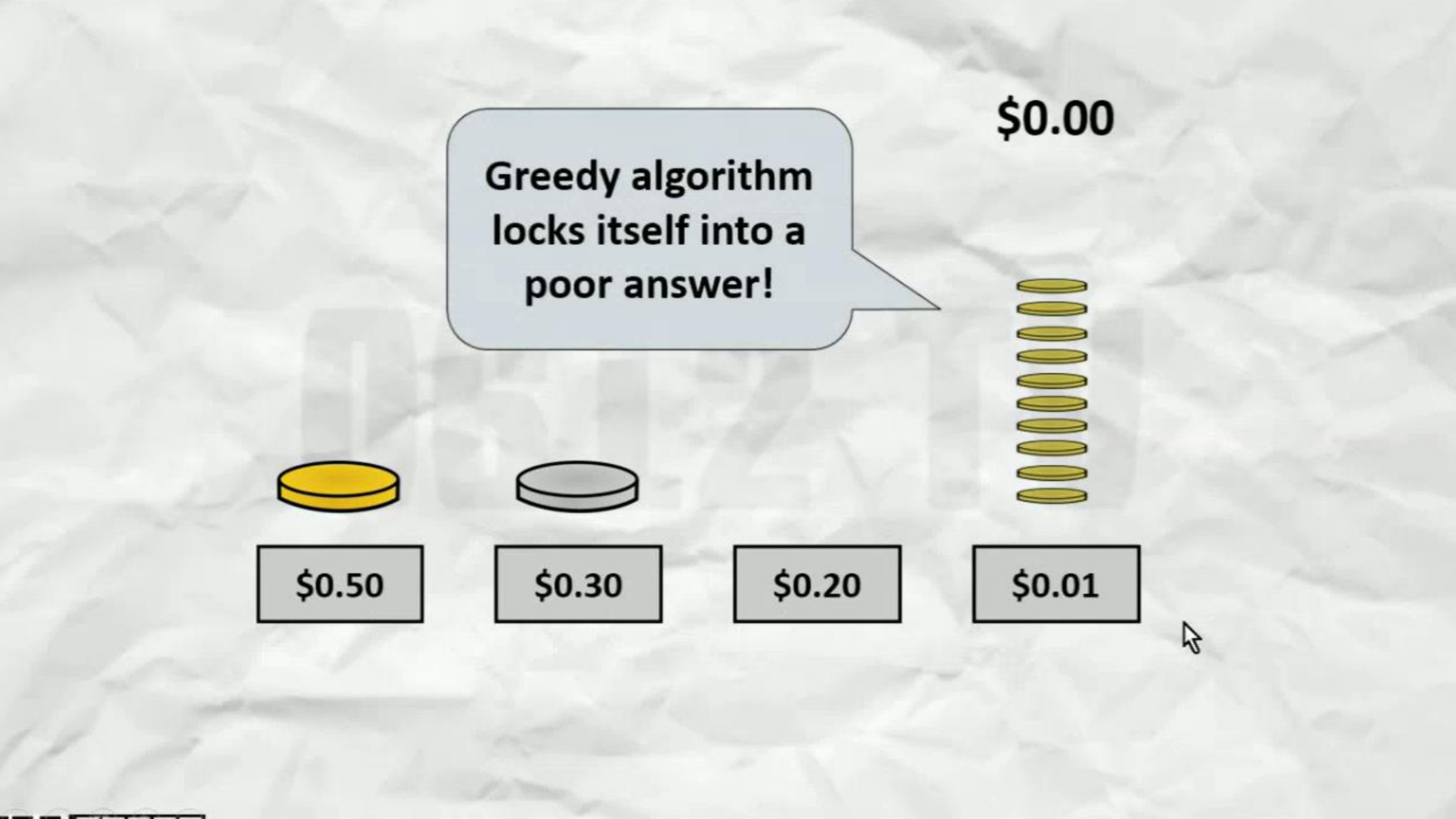
$0.30

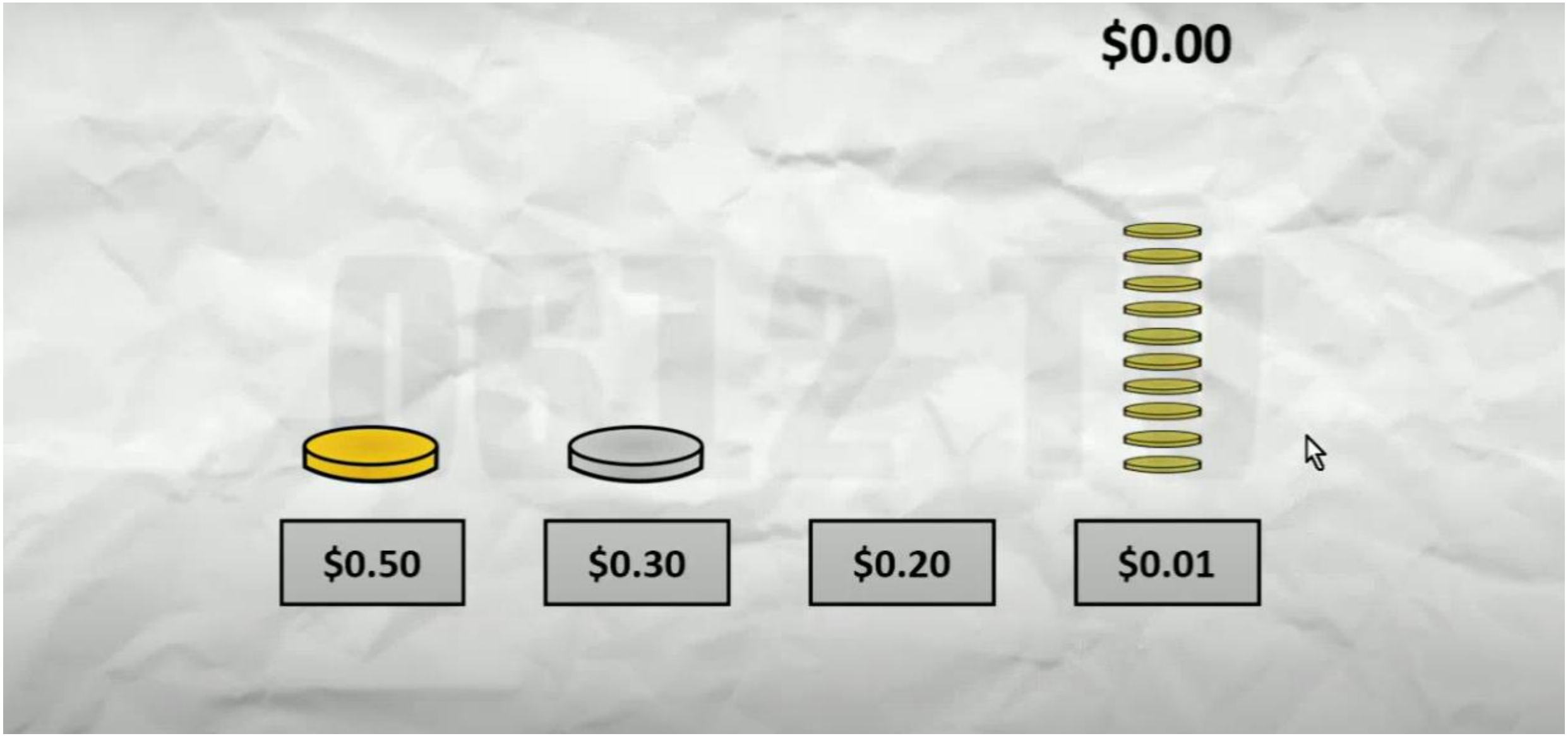
$0.20

$0.01

$0.10

$0.50	$0.30	$0.20	$0.01


$0.00
Greedy algorithm locks itself into a poor answer!
$0.50
$0.30
$0.20
$0.01


$0.00
$0.50
$0.30
$0.20
$0.01

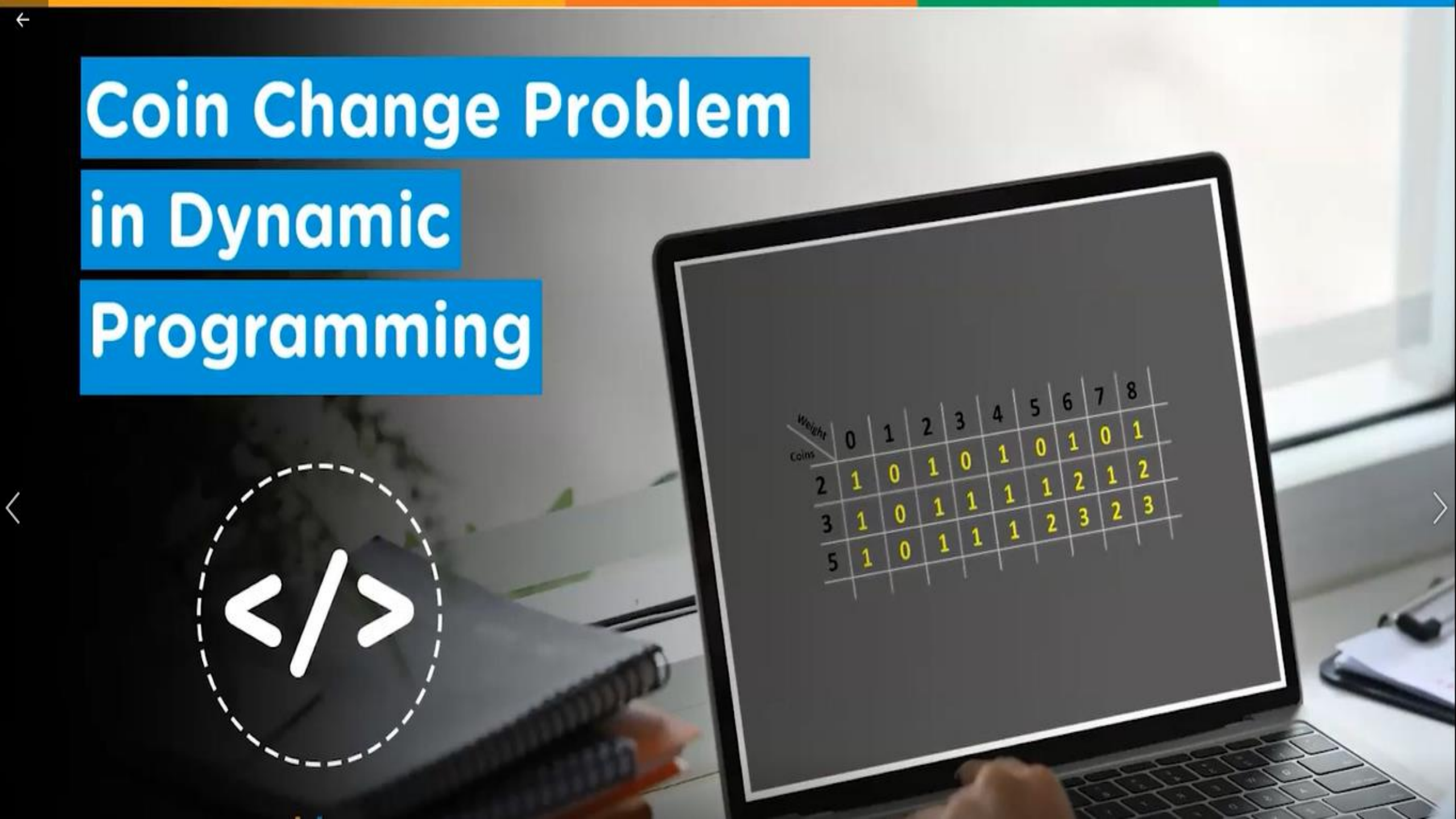
$0.90

$0.50 $0.30 $0.20 $0.01

$0.90

$0.50	$0.30	$0.20	$0.01

Coin Change Problem in Dynamic Programming



Agenda

What is Coin Change Problem?

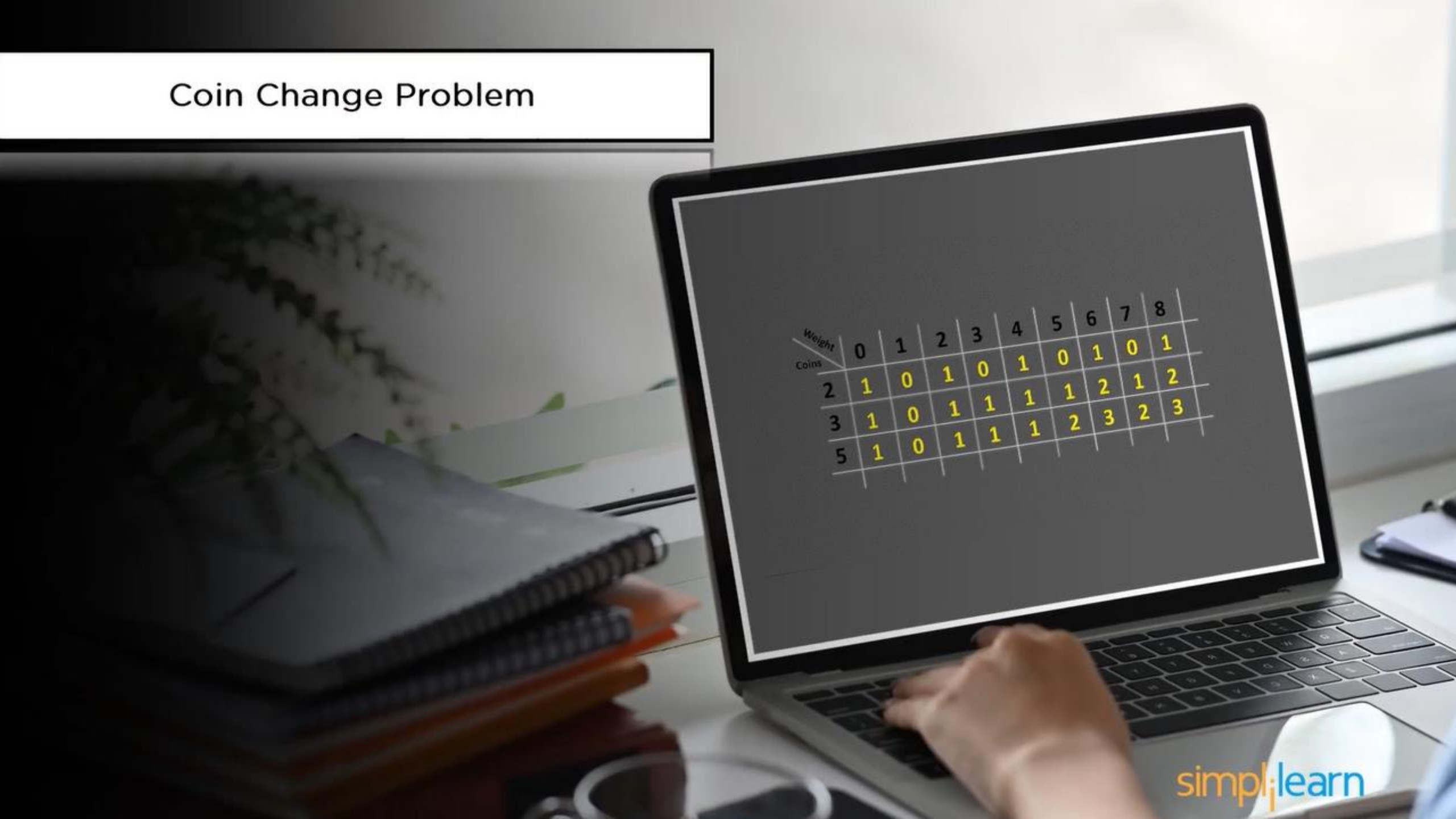
Agenda

Coin Change Problem Using Dynamic Programming

Agenda

Coin Change Problem Steps

Coin Change Problem



Coin Change Problem

Problem : Find the total number of ways you can change the given amount of money using the given coins.

Given : Different types of coins (OR) coins of varying denominations a specific amount (W).

Coin Change Problem

Coins

1

2

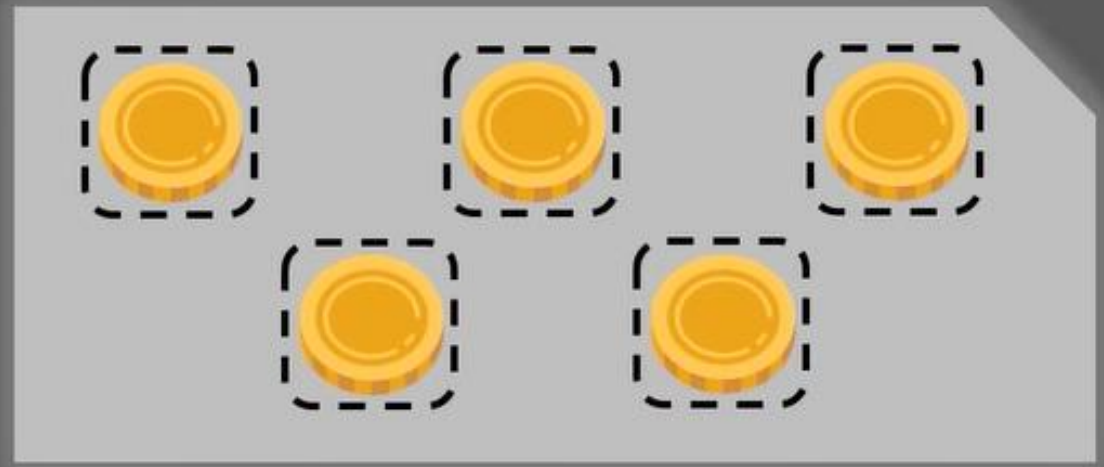
3

Weight
5

Condition : There will be an infinite supply of coins.

Coin Change Problem

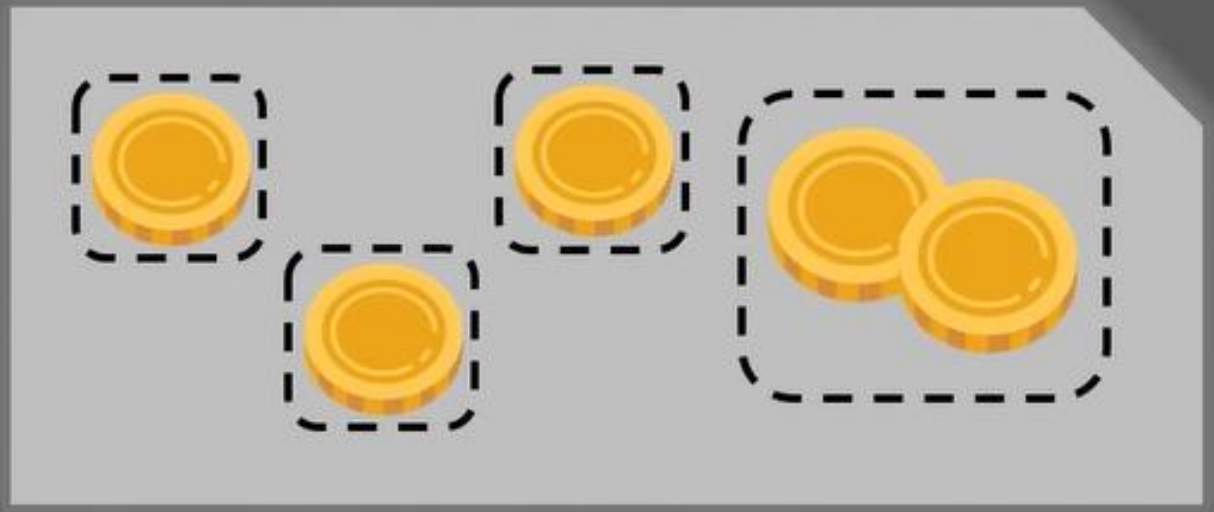
Number of Ways are:



Five one-rupee coin.

Coin Change Problem

Number of Ways are:



Three one-rupee coin and one two-rupee coin.

Coin Change Problem

Number of Ways are:

One one-rupee coin and two two-rupee coin.

Coin Change Problem

Number of Ways are:

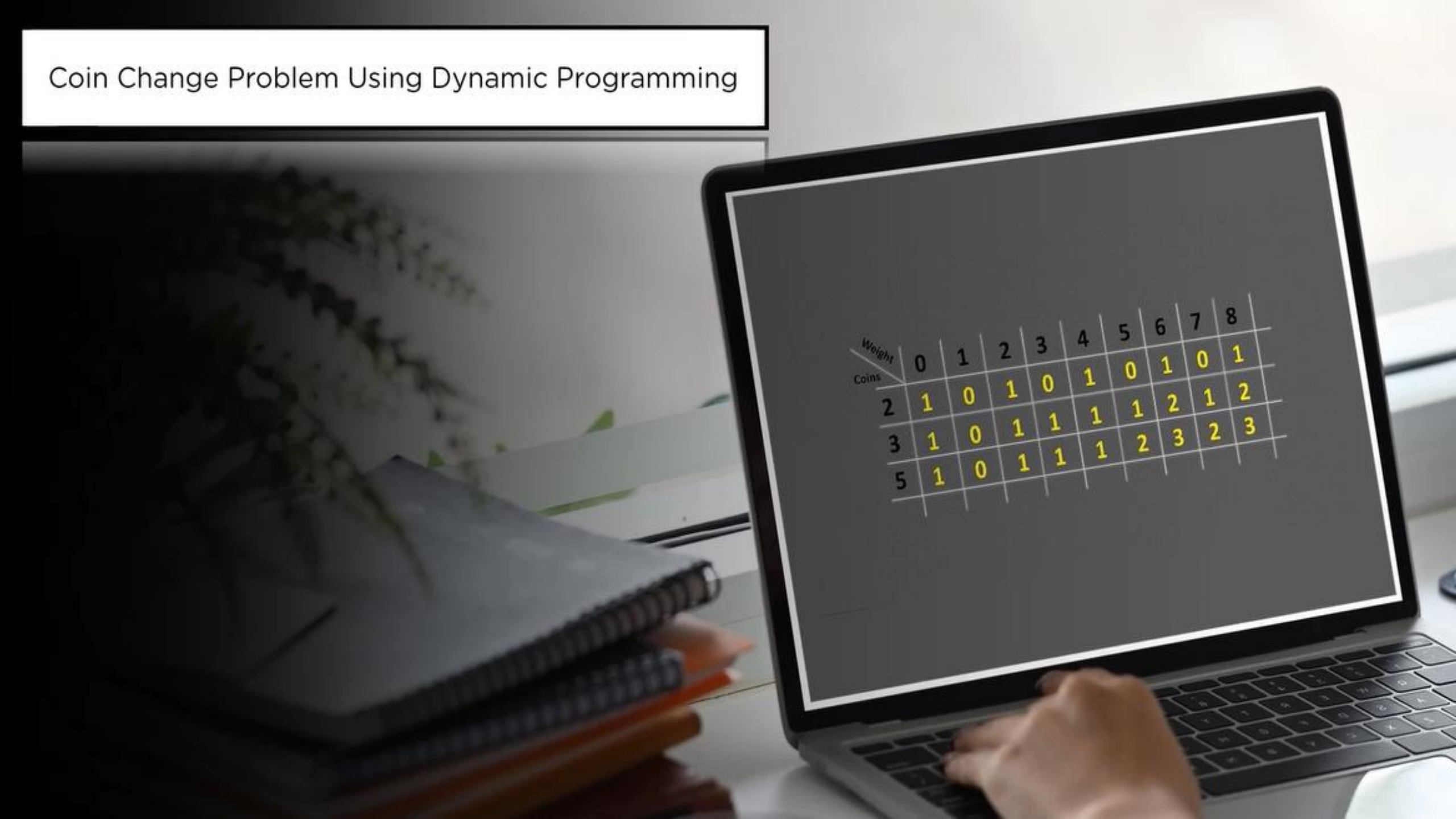
Two one-rupee coin and one three-rupee coin.

Coin Change Problem

Number of Ways are:

One two-rupee coin and one three-rupee coin.

Coin Change Problem Using Dynamic Programming



Coin Change Problem Using Dynamic Programming

Coins = { 2 , 3 , 5 }
Weight = 8

Coins \ Weight	0	1	2	3	4	5	6	7	8
2									
3									
5									

Coin Change Problem Using Dynamic Programming

Steps to solve coin change problem using dynamic programming:

1. Exclude the coin
2. Include the coin
3. Add step 1 and 2

Coin Change Problem Using Dynamic Programming

Coins \ Weight	0	1	2	3	4	5	6	7	8
2									
3									
5									

Coin Change Problem Using Dynamic Programming

Coins \ Weight	0	1	2	3	4	5	6	7	8
2	1								
3	1								
5	1								

Coin Change Problem Using Dynamic Programming

Coins \ Weight	0	1	2	3	4	5	6	7	8
2	1	0							
3	1								
5	1								

Coin Change Problem Using Dynamic Programming

Coins \ Weight	0	1	2	3	4	5	6	7	8
2	1	0	1						
3	1								
5	1								

Coin Change Problem Using Dynamic Programming

Coins \ Weight	0	1	2	3	4	5	6	7	8
2	1	0	1	0					
3	1								
5	1								

Coin Change Problem Using Dynamic Programming

Coins \ Weight	0	1	2	3	4	5	6	7	8
2	1	0	1	0	1				
3	1								
5	1								

Coin Change Problem Using Dynamic Programming

Coins \ Weight	0	1	2	3	4	5	6	7	8
2	1	0	1	0	1	0	1	0	1
3	1								
5	1								

Coin Change Problem Using Dynamic Programming

Coins \ Weight	0	1	2	3	4	5	6	7	8
2	1	0	1	0	1	0	1	0	1
3	1	0							
5	1								

If coin value > weight, then copy the above value

3 > 1

Coin Change Problem Using Dynamic Programming

Coins \ Weight	0	1	2	3	4	5	6	7	8
2	1	0	1	0	1	0	1	0	1
3	1	0	1						
5	1								

If coin value > weight, then copy the above value

3 > 2

Coin Change Problem Using Dynamic Programming

Coins \ Weight	0	1	2	3	4	5	6	7	8
2	1	0	1	0	1	0	1	0	1
3	1	0	1	0+1=1					
5	1								

Exclude coin 3 (copy the above value of coin 2 = 0) + include coin 3 (weight of 3 – coin value 3 = 0 and weight of 0 at coin value 3)

Coin Change Problem Using Dynamic Programming

Coins \ Weight	0	1	2	3	4	5	6	7	8
2	1	0	1	0	1	0	1	0	1
3	1	0	1	0+1=1	1+0=1				
5	1								

Coin Change Problem Using Dynamic Programming

Coins \ Weight	0	1	2	3	4	5	6	7	8
2	1	0	1	0	1	0	1	0	1
3	1	0	1	0+1=1	1+0=1				
5	1								

Exclude coin 3 (copy the above value of coin 2 = 1) + include coin 3 (weight of 4 – coin value 3 = 1 and weight of 1 at coin value 3)

Coin Change Problem Using Dynamic Programming

Coins \ Weight	0	1	2	3	4	5	6	7	8
2	1	0	1	0	1	0	1	0	1
3	1	0	1	0+1=1	1+0=1	1	2	1	2
5	1								

Coin Change Problem Using Dynamic Programming

Coins \ Weight	0	1	2	3	4	5	6	7	8
2	1	0	1	0	1	0	1	0	1
3	1	0	1	0+1=1	1+0=1	1	2	1	2
5	1								

Coin Change Problem Using Dynamic Programming

Coins \ Weight	0	1	2	3	4	5	6	7	8
2	1	0	1	0	1	0	1	0	1
3	1	0	1	0+1=1	1+0=1	1	2	1	2
5	1	0							

If coin value > weight, then copy the above value
5 > 1

Coin Change Problem Using Dynamic Programming

Coins \ Weight	0	1	2	3	4	5	6	7	8
2	1	0	1	0	1	0	1	0	1
3	1	0	1	0+1=1	1+0=1	1	2	1	2
5	1	0	1						

If coin value > weight, then copy the above value
5 > 2

Coin Change Problem Using Dynamic Programming

Coins \ Weight	0	1	2	3	4	5	6	7	8
2	1	0	1	0	1	0	1	0	1
3	1	0	1	0+1=1	1+0=1	1	2	1	2
5	1	0	1	1					

If coin value > weight, then copy the above value
5 > 3

Coin Change Problem Using Dynamic Programming

Coins \ Weight	0	1	2	3	4	5	6	7	8
2	1	0	1	0	1	0	1	0	1
3	1	0	1	0+1=1	1+0=1	1	2	1	2
5	1	0	1	1	1				

If coin value > weight, then copy the above value

5~~>5~~

Coin Change Problem Using Dynamic Programming

Coins \ Weight	0	1	2	3	4	5	6	7	8
2	1	0	1	0	1	0	1	0	1
3	1	0	1	0+1=1	1+0=1	1	2	1	2
5	1	0	1	1	1	1+1=2			

Exclude coin 5 (copy the above value of coin 2 and 3 = 0 +1=1) + include coin 5 (weight of 5 – coin value 5 = 0 and weight of 0 at coin value 5)

Coin Change Problem Using Dynamic Programming

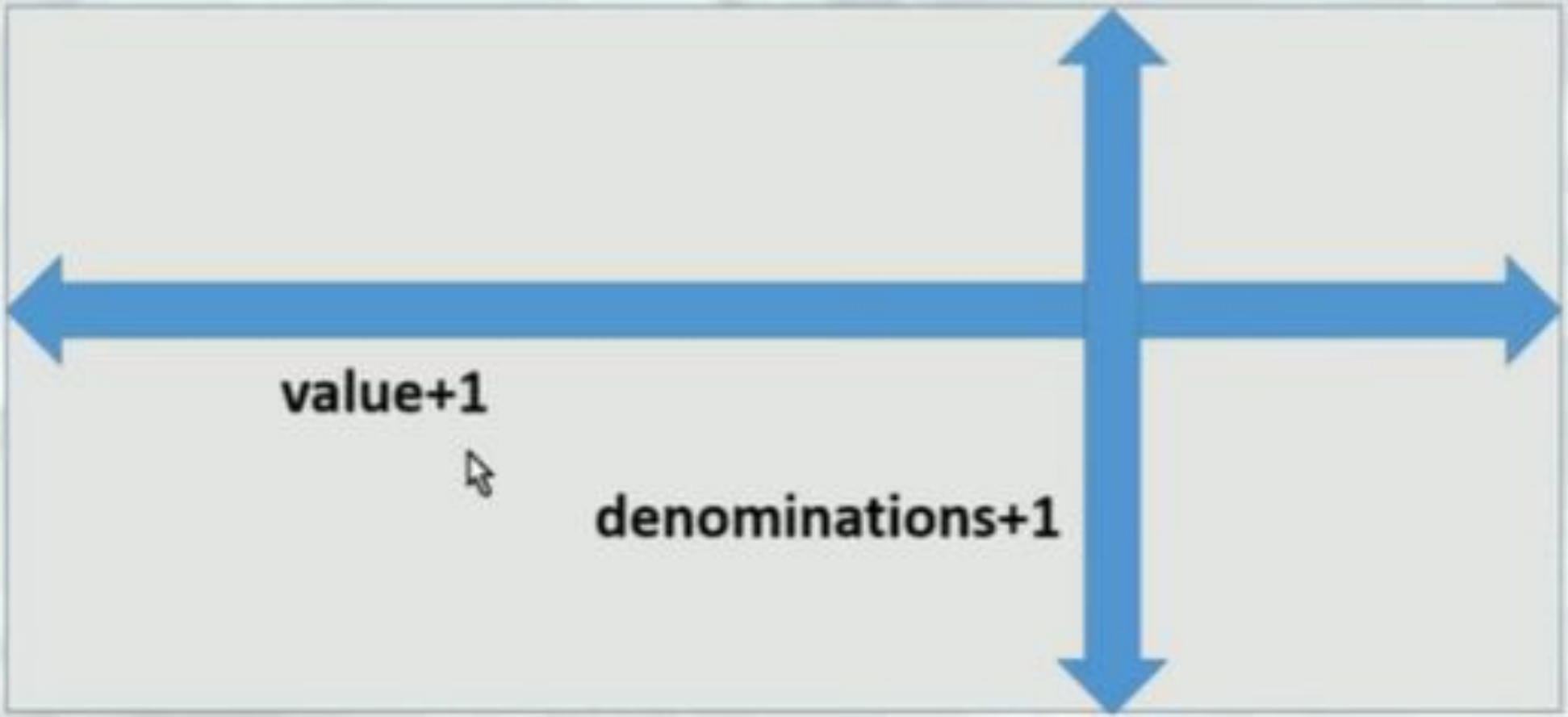
Coins \ Weight	0	1	2	3	4	5	6	7	8
2	1	0	1	0	1	0	1	0	1
3	1	0	1	1	1	1	2	1	2
5	1	0	1	1	1	2	3	2	4

Coin Change Problem Steps

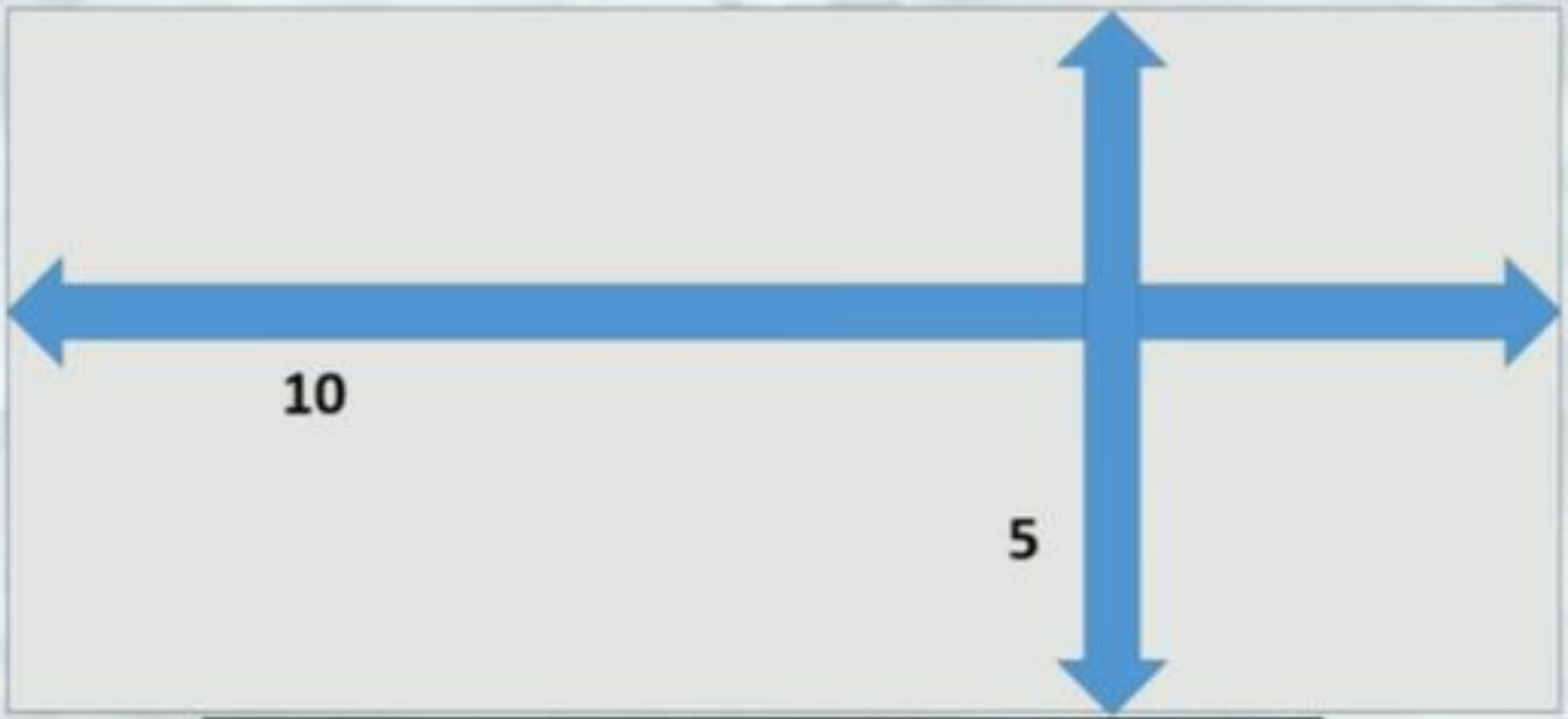
coin_change[i][j] // coin change is the name of matrix i is denotes the value of coin and j denotes the weight

```
for( i =0 ; i <= coins.length ; i++ )
 {
   for ( j =0 ; j <=weight ; j++ )
    {
      if ( coins[i] > j )
        coin_change[i][j] = coin_change[i-1][j]
      else
         coin_change[i][j]= coin_change[i-1][j] + coin_change[i][j-
coins[i]
     }
   }
```

Memoization Technique



Memoization Technique




```
1  denominations = [1, 2, 3, 5]
2  changeAmount = 9
```

$$\{0,1,2\} \quad = 3$$

Coins / Denomination---→	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						

Coins / Denomination---→	0	1	2	3	4	5
0	1	0	0	0	0	0
1	1	0 + 1	0+1	0+1	0+1	0+1
2	1	1	1+1	1+1	1+2	1+2
3	1	1	2	2+1	3+1	3 + 2
4	1	1	2	3	4 + 1	5 + 1
5	1	1	2	3	5	6 + 1

Coins / Denomination---→	0	1	2	3	4	5
0	1	0	0	0	0	0
1	1	1	1	1	1	1
2	1	1	2	2	3	3
3	1	1	2	3	4	5
4	1	1	2	3	5	6
5	1	1	2	3	5	7