Question Summary :

[pre requsite]

You re given an value “N” and the array of values [1 , 2 , 4 ,6]

Find the number of ways to form the number “N” by using the above number .

Observation :

Number of ways to create 1 → 1

Number of ways to create 2 → 2 (1 + 1 , 2)

Number of ways to create 3 → 3 (1 + 1 + 1 , 1+ 2 , 2 + 1)

Number of ways to create 4 → 6 (1 + 1+ 1+1 , 1+ 1 + 2 , 1 + 2 + 1 , 2 + 1 + 1 , 2 + 2 , 4)

By observing 3 , Number of ways that finished by 1 at the last (2) + number of ways 2 at the last (1) = 3 dp[3 - 1 ] + dp[3 - 2]

By observing 4 , Number of ways 1 finished last (3 ) + number of ways 2 finished last (2) + number of ways 4 finished last (1) = 6

Dp[4 - 1] + dp[4 - 2] + dp[4 - 4]

5 → number of ways 1 finisied last + number of ways 2 finished last + number of ways 4 finished last

Dp[5 - 1] + dp[5 - 2] + dp[5 - 4] = 6 + 3 + 1 → 10

* We observe that dp[0] & dp[1] is 1.

Recurrence Relation :

Dp[i] = d[i - 1] + dp[i - 2] + dp[i - 4] + dp[i - 6]

Code :

class Solution {

//coins == [1 , 2 , 4 , 6]

public int coinChange(int[] coins, int n) {

int [] dp = new int[n + 1];

dp[0] = dp[1] = 1;

dp[2] = 2;

for(int i = 3 ; i <=n ; i++){

dp[i] = dp[i - 1] + dp[i - 2];

if(i >= 4){

dp[i] += dp[i - 4];

}

if(i >= 6){

dp[i] += dp[i - 6];

}

}

return dp[n];

}}

Actual Question : Find the number of ways to create “N” by using 4 atmost 2 times

Observation :

Atmost 2 means , 0 , 1 ,or 2

So we need to maintain states for different types of 4 occurrence

An amount “i” can be formed

* By using “0” 4 coins
* By using “1” 4 coins
* By using “2” 4 coins

So i need to maintain three different state and print the sum as final answer.

Recurrence relation :

Dp[i][0] = d[i - 1][0] + dp[i - 2][0] + dp[i - 6] [0] // No four is used

Dp[i][1] = dp[i - 1][1] + dp[1 - 2][1] + dp[i - 4][0] + dp[i - 6][1]

Why dp[i - 4][0] ?

Because , it already included “4” , so there is no need to include the 4

Dp[i][2] = dp[i - 1][2] + dp[1 - 2][2] + dp[i - 4][1] + dp[i - 6][2]

Code :

class Solution {

//coins == [1 , 2 , 4 , 6]

public int coinChange(int[] coins, int n) {

int [][] dp = new int[n + 1][3];

dp[0][0] = 1;

for(int i = 1 ; i <=n ; i++){

if (i - 1 >= 0) dp[i][0] += dp[i - 1][0];

if (i - 2 >= 0) dp[i][0] += dp[i - 2][0];

if (i - 6 >= 0) dp[i][0] += dp[i - 6][0];

if (i - 1 >= 0) dp[i][1] += dp[i - 1][1];

if (i - 2 >= 0) dp[i][1] += dp[i - 2][1];

if (i - 4 >= 0) dp[i][1] += dp[i - 4][0];

if (i - 6 >= 0) dp[i][1] += dp[i - 6][1];

if (i - 1 >= 0) dp[i][2] += dp[i - 1][2];

if (i - 2 >= 0) dp[i][2] += dp[i - 2][2];

if (i - 4 >= 0) dp[i][2] += dp[i - 4][1];

if (i - 6 >= 0) dp[i][2] += dp[i - 6][2];

}

return dp[n][0] + dp[n][1] + dp[n][2];

}

}