

Date : / /

Coin Change II

Given an integer array coins representing different denominations & an integer amount representing a total amount of money.

Return no. of combinations that make up that amount.

Ex:- I/P : amount = 5, coins = [1, 2, 5]

O/P : 4

Explanation: $5 = 5$

$$5 = 2 + 2 + 1$$

$$5 = 2 + 1 + 1 + 1$$

$$5 = 1 + 1 + 1 + 1 + 1$$

Code:-

```
class Solution {  
    public int change(int amount, int[] coins) {  
        return f(0, amount, coins);  
    }  
  
    public int f(int ind, int amo, int[] coins) {  
        if (ind >= coins.length) { return amo == 0 ? 1 : 0; }  
        if (amo < 0) return 0;  
        int nottake = f(ind + 1, amo, coins);  
        int take = f(ind, amo - coins[ind], coins);  
        return take + nottake;  
    }  
}
```

Date : / /

~~Tabulation~~ Memoization

```
public int change(int amount, int[] coins) {  
    int[][] dp = new int[coins.length+1][amount+1];  
    for (int i=0; i<=coins.length; i++) {  
        Arrays.fill(dp[i], -1);  
    }  
    return f(0, amount, coins, dp);  
}
```

```
public int f(int ind, int amo, int[] coins, int[][] dp) {  
    if (ind >= coins.length) {  
        return amo == 0 ? 1 : 0;  
    }  
    if (dp[ind][amo] != -1) return dp[ind][amo];  
    int nottake = f(ind+1, amo, coins, dp);  
    int take = 0;  
    if (amo - coins[ind] >= 0) {  
        take = f(ind, amo - coins[ind], coins, dp);  
    }  
    dp[ind][amo] = take + nottake;  
    return dp[ind][amo];  
}
```

Date : / /

Tabulation

```
public int change (int amount, int[] coins) {  
    int n = coins.length;  
    int[][] dp = new int[n+1][amount+1];  
    dp[n][0] = 1;  
    for (int a=1; a <= amount; a++) { dp[n][a] = 0; }  
    for (int ind = n-1; ind >= 0; ind--) {  
        for (int amo = 0; amo <= amount; amo++) {  
            int nottake = dp[ind+1][amo];  
            int take = 0;  
            if (amo >= coins[ind]) { take = dp[ind][amo - coins[ind]]; }  
            dp[ind][amo] = take + nottake;  
        }  
    }  
    return dp[0][amount];  
}
```

Date : / /

Space Optimization

```
int n = coins.length;
int [] prev = new int[amount+1];
int [] curr = new int[amount+1];
prev[0] = 1;
for (int a = 1; a <= amount; a++) {
    prev[a] = 0;
}
for (int ind = n-1; ind >= 0; ind--) {
    for (int amo = 0; amo <= amount; amo++) {
        int noftake = prev[amo];
        int take = 0;
        if (amo >= coins[ind]) {
            take = curr[amo - coins[ind]];
        }
        curr[amo] = take + noftake;
    }
    prev = curr;
    curr = new int[amount+1];
}
```