Previous Next

518. Coin Change II

Nov. 16, 2019 | 27.9K views

Average Rating: 4.48 (48 votes)

You are given coins of different denominations and a total amount of money. Write a function to compute the number of combinations that make up that amount. You may assume that you have infinite number of each kind of coin.

Example 1:

```
Input: amount = 5, coins = [1, 2, 5]
Output: 4
Explanation: there are four ways to make up the amount:
5=5
5=2+2+1
5=2+1+1+1
5=1+1+1+1+1
```

Example 2:

```
Input: amount = 3, coins = [2]
Output: 0
Explanation: the amount of 3 cannot be made up just with coins of 2.
```

Example 3:

```
Input: amount = 10, coins = [10]
Output: 1
```

Note:

You can assume that

- 0 <= amount <= 5000
- 1 <= coin <= 5000
- the number of coins is less than 500
- the answer is guaranteed to fit into signed 32-bit integer

Solution

Approach 1: Dynamic Programming

Template

This is a classical dynamic programming problem.

Here is a template one could use:

- Define the base cases for which the answer is obvious.
- Develop the strategy to compute more complex case from more simple one.
- Link the answer to base cases with this strategy.

Example

Let's pic up an example: amount = 11, available coins - 2 cent, 5 cent and 10 cent. Note, that coins are unlimited.

Number of combinations that make up 11

Base Cases: No Coins or Amount = 0

If the total amount of money is zero, there is only one combination: to take zero coins.

Another base case is no coins: zero combinations for `amount > 0` and one combination for `amount == 0`.

Base cases

amount =	0	1	2	3	4	5	6	7	8	9	10	11
combinations using no coins =	1	0	0	0	0	0	0	0	0	0	0	0

2 Cent Coins

Let's do one step further and consider the situation with one kind of available coins: 2 cent.

amount =	0	1	2	3	4	5	6	7	8	9	10	11
combinations using no coins =	1	0	0	0	0	0	0	0	0	0	0	0
combinations with 2 cents =	1	0	1	0	1	0	1	0	1	0	1	0

It's quite evident that there could be 1 or 0 combinations here, 1 combination for even amount and 0 combinations for the odd one.

> The same answer could be received in a recursive way, by computing the number of combinations for all amounts of money, from 0 to 11.

First, that's quite obvious that all amounts less than 2 are *not* impacted by the presence of 2 cent coins. Hence for `amount = 0` and for `amount = 1` one could reuse the results from the figure 2.

Starting from `amount = 2`, one could use 2 cent coins in the combinations. Since the amounts are considered gradually from 2 to 11, at each given moment one could be sure to add not more than one coin to the previously known combinations.

So let's pick up 2 cent coin, and use it to make up `amount = 2`. The number of combinations with this 2 cent coin is a number combinations for `amount = 0`, i.e. 1.

amount =	0	1	2	3	4	5	6	7	8	9	10	11
combinations using no coins =	1	0	0	0	0	0	0	0	0	0	0	0
combinations with 2 cents =	1	0	1									

number of combinations for [amount = 2] =
number of current combinations for [amount = 2] +
number of current combinations for [amount - 2 = 0]
= 0 + 1 = 1

Now let's pick up 2 cent coin, and use it to make up `amount = 3`. The number of combinations with this 2 cent coin is a number combinations for `amount = 1`, i.e. 0.

amount =	0	1	2	3	4	5	6	7	8	9	10	11
combinations using no coins =	1	0	0	0	0	0	0	0	0	0	0	0
combinations with 2 cents =	1	0	1	0								

number of combinations for [amount = 3] =
number of current combinations for [amount = 3] +
number of current combinations for [amount - 2 = 1]
= 0 + 0 = 0

That leads to DP formula for number of combinations to make up the `amount = x`: `dp[x] = dp[x] + dp[x - coin]`, where `coin = 2 cents` is a value of coins we're currently adding.

amount =	0	1	2	3	4	5	6	7	8	9	10	11
combinations using no coins =	1	0	0	0	0	0	0	0	0	0	0	0
combinations with 2 cents =	1	0	1	0	1	0	1	0	1	0	1	0

dp[x] = dp[x] + dp[x - coin]
coin = 2 cents

2 Cent Coins + 5 Cent Coins + 10 Cent Coins

Now let's add 5 cent coins. The formula is the same, but do not forget to add `dp[x]`, number of combinations with 2 cent coins.

amount =	0	1	2	3	4	5	6	7	8	9	10	11
combinations using no coins =	1	0	0	0	0	0	0	0	0	0	0	0
combinations with 2 cents =	1	0	1	0	1	0	1	0	1	0	1	0
combinations with 2 and 5 cents =	1	0	1	0	1	1	1	1	1	1	2	1

dp[x] for x < coin = 5 cents are not going to change

dp[x] = dp[x] + dp[x - coin]
coin = 5 cents

The story is the same for 10 cent coins.

amount =	0	1	2	3	4	5	6	7	8	9	10	11
combinations using no coins =	1	0	0	0	0	0	0	0	0	0	0	0
combinations with 2 cents =	1	0	1	0	1	0	1	0	1	0	1	0
combinations with 2 and 5 cents =	1	0	1	0	1	1	1	1	1	1	2	1
combinations with 2, 5, and 10 cents =	1	0	1	0	1	1	1	1	1	1	3	1

dp[x] for x < coin = 10 cents are not going to change

dp[x] = dp[x] + dp[x - coin]
coin = 10 cents

Now the strategy is here:

- Add coins one-by-one, starting from base case "no coins".
- For each added coin, compute recursively the number of combinations for each amount of money from 0 to `amount`.

Algorithm

- Initiate number of combinations array with the base case "no coins": `dp[0] = 1`, and all the rest = 0.
- Loop over all coins:
  - For each coin, loop over all amounts from 0 to `amount`:
    - For each amount x, compute the number of combinations: `dp[x] += dp[x - coin]`.
- Return `dp[amount]`.

Implementation

Java | Python — Copy

```python
class Solution:
    def change(self, amount: int, coins: List[int]) -> int:
        dp = [0] * (amount + 1)
        dp[0] = 1

        for coin in coins:
            for x in range(coin, amount + 1):
                dp[x] += dp[x - coin]
        return dp[amount]
```

Complexity Analysis

- Time complexity: $\mathcal{O}(N \times \text{amount})$, where N is a length of coins array.
- Space complexity: $\mathcal{O}(\text{amount})$ to keep dp array.

Rate this article:

Previous Next

Comments: 19 Sort By

Type comment here... (Markdown is supported)

Preview Post

yolo31 ★ 183 April 2, 2020 12:46 PM
great article...it would be great if you guys can first explain top down approach..then bottom up
16 Share Reply
SHOW 1 REPLY

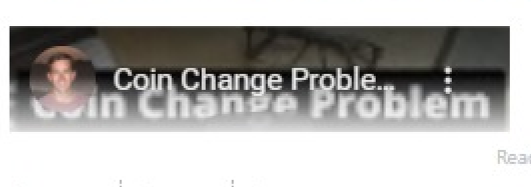
babhishek21 ★ 88 June 10, 2020 1:31 PM
The fun part about this solution is that if you switch the order of the `for` loops in the code, your answer almost doubles! Learnt it the hard way.

If you're wondering why, here's a hint:

Read More
6 Share Reply
SHOW 1 REPLY

he11mean ★ 12 June 8, 2020 12:58 AM
Although the implementation looks super easy and so does the explanation neither of them is. Moreover, it is dangerously confusing. The dp relationship `dp[x] = dp[x] + dp[x - coin]` depends on itself, which is not permitted for dynamic programming. Optimal substructure is not shown. So this is indeed a solution, but it is not a dynamic programming one (strictly speaking at least). The problem can be solved with DP, however, for which you need a two dimensional DP array.
Read More
6 Share Reply
SHOW 2 REPLIES

egch ★ 12 June 8, 2020 2:18 AM
This video describe the problem in a great way https://www.youtube.com/watch?v=jaNZ83Q3QGc



Read More
5 Share Reply
SHOW 2 REPLIES

jkrishna ★ 5 February 1, 2020 9:16 PM
nice answer
5 Share Reply

kevin2702 ★ 141 June 8, 2020 6:26 AM
I would say the dp formula `dp[x] += dp[x - coin]` can be explained as
The combination of i number of coins to make change for j amount = The combination when we not using `ith` coin to make change for j amount + The combination when we use `ith` coin to make change for j amount.
Read More
3 Share Reply
SHOW 1 REPLY

namidairo777 ★ 5 May 5, 2020 8:50 AM
DFS + backtracking can also solve it. But for large amount of inputs, it will fail due to Time Limit Exceeded 😑
4 Share Reply
SHOW 5 REPLIES

Zaizi ★ 2 March 16, 2020 1:19 PM
The corner case for this problem feels strange to me:

Wrong Answer
Your input: 0, []
Output: 0
Read More
2 Share Reply
SHOW 2 REPLIES

david2999999 ★ 49 June 14, 2020 7:19 PM
Brute Force Recursive. TLE error

```
public class CoinChangeRecursiveApproach {
    public int change(int amount, int[] coins) {
        return change(amount, coins, 0);
```

Read More
1 Share Reply
SHOW 1 REPLY

hardikchheda13 ★ 7 April 23, 2020 5:18 AM
What is the rationale behind making dp[0] = 1?
1 Share Reply
SHOW 4 REPLIES

‹ 1 2 ›