Coin Change II

Problem Statement:

You are given an integer array `coins` representing coins of different denominations and an integer `amount` representing a total amount of money.

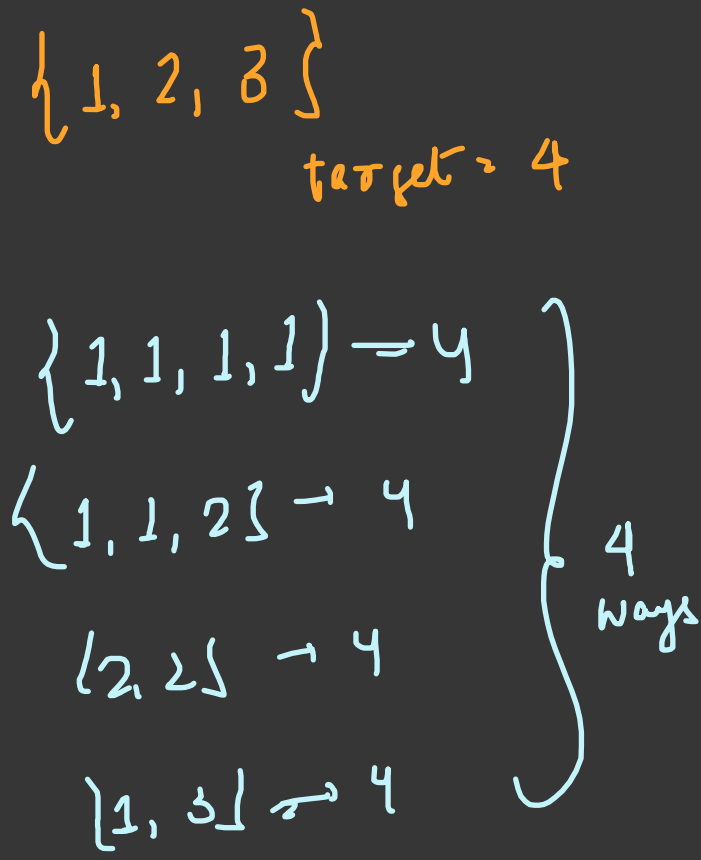
Return *the number of combinations that make up that amount*. If that amount of money cannot be made up by any combination of the coins, return `0`.

You may assume that you have an infinite number of each kind of coin.

The answer is **guaranteed** to fit into a signed **32-bit** integer.

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
```



We have find no. of ways. : Try out all ways

Recursion

For Total ways base case return 1 if condition met

Recursion:

1. Express → index f(ind, T)

2- Explore all possibility : → not take
→ take

3 Sum all possibility ~~con~~ and return.

Recursion function

f(ind, T)

if(ind == 20)

~~if(t == target)~~

return target % a[0] == 0;

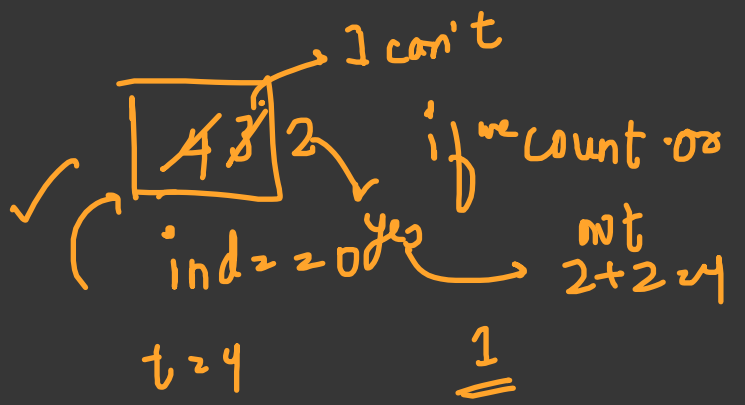
notTake = f(ind-1, T);

take = 0;

if(arr[ind] <= T)

take = f(ind, T - arr[ind]);

return take + notTake;



if target % arr[0]
then target can be
formed.

T.C. : $>>O(2^N)$ exponential

S.C. : O(N) → auxiliary space

Memoization.

Tabulation:

Space Optimisation:

} Similar to Minimum Coin DP with little change