LeetCode 518

https://leetcode.com/problems/coin-change-2/description/

Yifeng Zeng

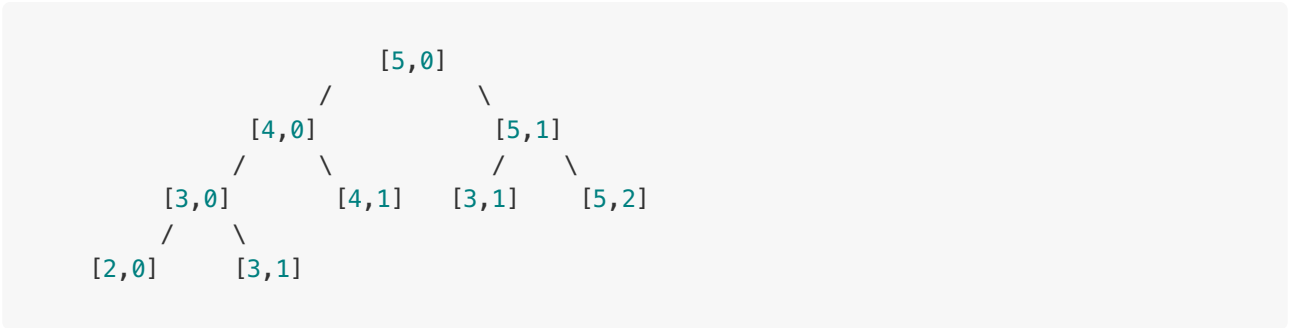
Description

518. Coin Change 2

Idea Report

Still, we can split it into two different situations. 1. choose the current coin, then the next recursion level I'm looking for amount - coins[index], and still looking at this index. 2. not choose the current coin, then the next recursion level I'm still looking for the amount, but the index become index + 1.

Take example [1,2,5] with amount 5. We define a pair [amount, index], then we can draw a solution space tree. Left child is to choose current index, and right child is not to choose current index. We can see that there might be a duplication, so we create a int[][] memo to do the memorized search.



Code

```
class Solution {
    public int change(int amount, int[] coins) {
        Arrays.sort(coins);
        int[][] memo = new int[amount + 1][coins.length + 1];
        for (int[] row : memo) {
            Arrays.fill(row, -1);
```

```
        }
        return helper(amount, coins, 0, memo);
    }

    private int helper(int amount, int[] coins, int index, int[][] memo) {
        if (amount == 0) {
            return 1;
        }
        if (amount < 0 || index >= coins.length) {
            return 0;
        }
        if (memo[amount][index] != -1) {
            return memo[amount][index];
        }

        int number = 0;
        // if we choose index
        if (amount - coins[index] >= 0) {
            number += helper(amount - coins[index], coins, index, memo);
        }
        // if we not choose index
        number += helper(amount, coins, index + 1, memo);
        return memo[amount][index] = number;
    }
}
```

Summary

- Try to create solution space tree which is really helpful
- When there is a duplication in the search, use something to save the values that has already been calculated.