Dp

动态一维：零钱兑换、解码方法、多米诺和托米诺平铺（重要）

473 火柴拼正方形—21年真题

1000 合并石头的最低成本

1125 最小必要团队

152乘积最大子数组

买股票的最佳时机

贪心：

11

45

122

盛水最多的容器

跳跃游戏II

买股票的最佳时机

134

179

376

397

410

435

455（今天可以做）

763

学习重构堆（优先队列）比较规则表达式

Unordered\_map的几个方法

滑动窗口题：

3

76

239

632

普通数组、二叉树：hot 100全刷

学习如何把一个数组建成一棵二叉树，数组🡪BST

Hot 100 堆就做前两道

Dfs

岛屿数量 最大岛屿数量

找dijstra模板

图

684

743

787

841

997

1514

1615（重要）

1971

1976

1383（最大团队表现值）重要

Priority\_queue的重构、贪心、堆优化

刷题一定要总结，可以2到4天后再回来看看自己还会不会

矩阵相乘学一下

双指针 + 二分查找

34

最小债

20

739

target 在外层保证了较小的target包括了所有的情况，再以每个数作为结尾，使得一个组合里的每个数都作为结尾的情况都算了一次解； target 在内层，每个数作用于每个 target，此时 target 还没有算完，后面可能还有些新的数可以 是什么意思

**《代码随想录》算法视频公开课：**[**装满背包有多少种方法？组合与排列有讲究！| LeetCode：518.零钱兑换II**](https://leetcode.cn/link/?target=https%3A%2F%2Fwww.bilibili.com%2Fvideo%2FBV1KM411k75j%2F)**，相信结合视频再看本篇题解，更有助于大家对本题的理解**。

**思路**

这是一道典型的背包问题，一看到钱币数量不限，就知道这是一个完全背包。

对完全背包还不了解的同学，可以看这篇：[动态规划：关于完全背包，你该了解这些！](https://leetcode.cn/link/?target=https%3A%2F%2Fprogrammercarl.com%2F%E8%83%8C%E5%8C%85%E9%97%AE%E9%A2%98%E7%90%86%E8%AE%BA%E5%9F%BA%E7%A1%80%E5%AE%8C%E5%85%A8%E8%83%8C%E5%8C%85.html)

但本题和纯完全背包不一样，**纯完全背包是凑成背包最大价值是多少，而本题是要求凑成总金额的物品组合个数！**

注意题目描述中是凑成总金额的硬币组合数，为什么强调是组合数呢？

例如示例一：

5 = 2 + 2 + 1

5 = 2 + 1 + 2

这是一种组合，都是 2 2 1。

如果问的是排列数，那么上面就是两种排列了。

**组合不强调元素之间的顺序，排列强调元素之间的顺序**。 其实这一点我们在讲解回溯算法专题的时候就讲过了哈。

那我为什么要介绍这些呢，因为这和下文讲解遍历顺序息息相关!

回归本题，动规五步曲来分析如下：

1. 确定dp数组以及下标的含义

dp[j]：凑成总金额j的货币组合数为dp[j]

1. 确定递推公式

dp[j] 就是所有的dp[j - coins[i]]（考虑coins[i]的情况）相加。

所以递推公式：dp[j] += dp[j - coins[i]];

**这个递推公式大家应该不陌生了，我在讲解01背包题目的时候在这篇**[**494. 目标和**](https://leetcode.cn/link/?target=https%3A%2F%2Fprogrammercarl.com%2F0494.%E7%9B%AE%E6%A0%87%E5%92%8C.html)**中就讲解了，求装满背包有几种方法，公式都是：dp[j] += dp[j - nums[i]];**

1. dp数组如何初始化

首先dp[0]一定要为1，dp[0] = 1是 递归公式的基础。如果dp[0] = 0 的话，后面所有推导出来的值都是0了。

那么 dp[0] = 1 有没有含义，其实既可以说 凑成总金额0的货币组合数为1，也可以说 凑成总金额0的货币组合数为0，好像都没有毛病。

但题目描述中，也没明确说 amount = 0 的情况，结果应该是多少。

这里我认为题目描述还是要说明一下，因为后台测试数据是默认，amount = 0 的情况，组合数为1的。

下标非0的dp[j]初始化为0，这样累计加dp[j - coins[i]]的时候才不会影响真正的dp[j]

dp[0]=1还说明了一种情况：如果正好选了coins[i]后，也就是j-coins[i] == 0的情况表示这个硬币刚好能选，此时dp[0]为1表示只选coins[i]存在这样的一种选法。

1. 确定遍历顺序

本题中我们是外层for循环遍历物品（钱币），内层for遍历背包（金钱总额），还是外层for遍历背包（金钱总额），内层for循环遍历物品（钱币）呢？

我在[动态规划：关于完全背包，你该了解这些！](https://leetcode.cn/link/?target=https%3A%2F%2Fprogrammercarl.com%2F%E8%83%8C%E5%8C%85%E9%97%AE%E9%A2%98%E7%90%86%E8%AE%BA%E5%9F%BA%E7%A1%80%E5%AE%8C%E5%85%A8%E8%83%8C%E5%8C%85.html)中讲解了完全背包的两个for循环的先后顺序都是可以的。

**但本题就不行了！**

因为纯完全背包求得装满背包的最大价值是多少，和凑成总和的元素有没有顺序没关系，即：有顺序也行，没有顺序也行！

而本题要求凑成总和的组合数，元素之间明确要求没有顺序。

所以纯完全背包是能凑成总和就行，不用管怎么凑的。

本题是求凑出来的方案个数，且每个方案个数是为组合数。

那么本题，两个for循环的先后顺序可就有说法了。

我们先来看 外层for循环遍历物品（钱币），内层for遍历背包（金钱总额）的情况。

代码如下：

for (int i = 0; i < coins.size(); i++) { // 遍历物品

for (int j = coins[i]; j <= amount; j++) { // 遍历背包容量

dp[j] += dp[j - coins[i]];

}

}

假设：coins[0] = 1，coins[1] = 5。

那么就是先把1加入计算，然后再把5加入计算，得到的方法数量只有{1, 5}这种情况。而不会出现{5, 1}的情况。

**所以这种遍历顺序中dp[j]里计算的是组合数！**

如果把两个for交换顺序，代码如下：

for (int j = 0; j <= amount; j++) { // 遍历背包容量

for (int i = 0; i < coins.size(); i++) { // 遍历物品

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

}

}

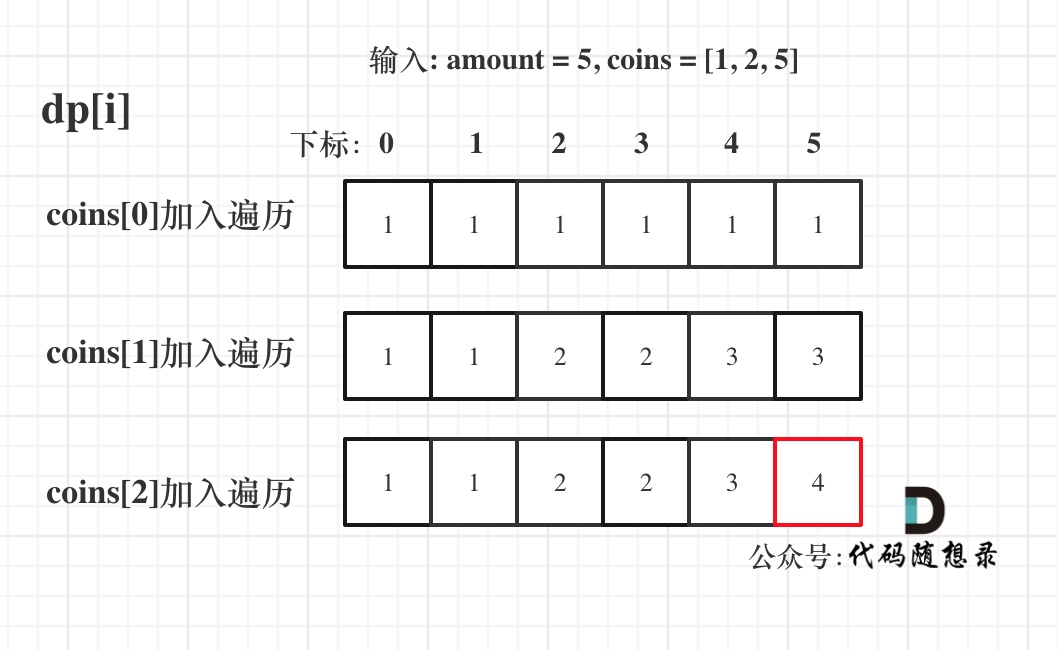
背包容量的每一个值，都是经过 1 和 5 的计算，包含了{1, 5} 和 {5, 1}两种情况。

**此时dp[j]里算出来的就是排列数！**

可能这里很多同学还不是很理解，**建议动手把这两种方案的dp数组数值变化打印出来，对比看一看！（实践出真知）**

1. 举例推导dp数组

输入: amount = 5, coins = [1, 2, 5] ，dp状态图如下：



最后红色框dp[amount]为最终结果。

以上分析完毕，C++代码如下：

class Solution {

public:

int change(int amount, vector<int>& coins) {

vector<int> dp(amount + 1, 0);

dp[0] = 1;

for (int i = 0; i < coins.size(); i++) { // 遍历物品

for (int j = coins[i]; j <= amount; j++) { // 遍历背包

dp[j] += dp[j - coins[i]];

}

}

return dp[amount];

}

};

是不是发现代码如此精简，哈哈

**总结**

本题的递推公式，其实我们在[494. 目标和](https://leetcode.cn/link/?target=https%3A%2F%2Fprogrammercarl.com%2F0494.%E7%9B%AE%E6%A0%87%E5%92%8C.html)中就已经讲过了，**而难点在于遍历顺序！**

在求装满背包有几种方案的时候，认清遍历顺序是非常关键的。

**如果求组合数就是外层for循环遍历物品，内层for遍历背包**。

**如果求排列数就是外层for遍历背包，内层for循环遍历物品**。

可能说到排列数录友们已经有点懵了，后面Carl还会安排求排列数的题目，到时候在对比一下，大家就会发现神奇所在！

**其他语言版本**

Java

Python

go

Rust

JavaScript

TypeScript

Scala

class Solution {

public int change(int amount, int[] coins) {

//递推表达式

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

//初始化dp数组，表示金额为0时只有一种情况，也就是什么都不装

dp[0] = 1;

for (int i = 0; i < coins.length; i++) {

for (int j = coins[i]; j <= amount; j++) {

dp[j] += dp[j - coins[i]];

}

}

return dp[amount];

}

}

**背包问题力扣完整攻略**

只要按如下顺序刷题，相信会帮你在学习背包问题的路上少走很多弯路！

* 0-1背包
  + [01背包问题理论基础二维dp数组](https://leetcode.cn/link/?target=https%3A%2F%2Fprogrammercarl.com%2F%E8%83%8C%E5%8C%85%E7%90%86%E8%AE%BA%E5%9F%BA%E7%A1%8001%E8%83%8C%E5%8C%85-1.html)
  + [01背包问题理论基础一维dp数组（滚动数组）](https://leetcode.cn/link/?target=https%3A%2F%2Fprogrammercarl.com%2F%E8%83%8C%E5%8C%85%E7%90%86%E8%AE%BA%E5%9F%BA%E7%A1%8001%E8%83%8C%E5%8C%85-2.html)
  + [416.分割等和子集](https://leetcode.cn/link/?target=https%3A%2F%2Fprogrammercarl.com%2F0416.%E5%88%86%E5%89%B2%E7%AD%89%E5%92%8C%E5%AD%90%E9%9B%86.html)
  + [1049.最后一块石头的重量 II](https://leetcode.cn/link/?target=https%3A%2F%2Fprogrammercarl.com%2F1049.%E6%9C%80%E5%90%8E%E4%B8%80%E5%9D%97%E7%9F%B3%E5%A4%B4%E7%9A%84%E9%87%8D%E9%87%8FII.html)
  + [494.目标和](https://leetcode.cn/link/?target=https%3A%2F%2Fprogrammercarl.com%2F0494.%E7%9B%AE%E6%A0%87%E5%92%8C.html)
  + [474.一和零](https://leetcode.cn/link/?target=https%3A%2F%2Fprogrammercarl.com%2F0474.%E4%B8%80%E5%92%8C%E9%9B%B6.html)
* 完全背包
  + [动态规划：关于完全背包，你该了解这些！](https://leetcode.cn/link/?target=https%3A%2F%2Fprogrammercarl.com%2F%E8%83%8C%E5%8C%85%E9%97%AE%E9%A2%98%E7%90%86%E8%AE%BA%E5%9F%BA%E7%A1%80%E5%AE%8C%E5%85%A8%E8%83%8C%E5%8C%85.html)
  + [518. 零钱兑换 II](https://leetcode.cn/link/?target=https%3A%2F%2Fprogrammercarl.com%2F0518.%E9%9B%B6%E9%92%B1%E5%85%91%E6%8D%A2II.html)
  + [0377.组合总和Ⅳ.](https://leetcode.cn/link/?target=https%3A%2F%2Fprogrammercarl.com%2F0377.%E7%BB%84%E5%90%88%E6%80%BB%E5%92%8C%E2%85%A3.html)
  + [70.爬楼梯进阶版](https://leetcode.cn/link/?target=https%3A%2F%2Fprogrammercarl.com%2F0070.%E7%88%AC%E6%A5%BC%E6%A2%AF%E5%AE%8C%E5%85%A8%E8%83%8C%E5%8C%85%E7%89%88%E6%9C%AC.html)
  + [322. 零钱兑换](https://leetcode.cn/link/?target=https%3A%2F%2Fprogrammercarl.com%2F0322.%E9%9B%B6%E9%92%B1%E5%85%91%E6%8D%A2.html)
  + [279.完全平方数](https://leetcode.cn/link/?target=https%3A%2F%2Fprogrammercarl.com%2F0279.%E5%AE%8C%E5%85%A8%E5%B9%B3%E6%96%B9%E6%95%B0.html)
  + [139.单词拆分](https://leetcode.cn/link/?target=https%3A%2F%2Fprogrammercarl.com%2F0139.%E5%8D%95%E8%AF%8D%E6%8B%86%E5%88%86.html)
* 多重背包
  + [动态规划：关于多重背包，你该了解这些！](https://leetcode.cn/link/?target=https%3A%2F%2Fprogrammercarl.com%2F%E8%83%8C%E5%8C%85%E9%97%AE%E9%A2%98%E7%90%86%E8%AE%BA%E5%9F%BA%E7%A1%80%E5%A4%9A%E9%87%8D%E8%83%8C%E5%8C%85.html)
* [背包问题总结篇](https://leetcode.cn/link/?target=https%3A%2F%2Fprogrammercarl.com%2F%E8%83%8C%E5%8C%85%E6%80%BB%E7%BB%93%E7%AF%87.html)

大家好，我是程序员Carl，如果你还在没有章法的刷题，建议按照[代码随想录刷题路线](https://leetcode.cn/link/?target=https%3A%2F%2Fprogrammercarl.com%2F)来刷，并提供[PDF下载](https://leetcode.cn/link/?target=https%3A%2F%2Fprogrammercarl.com%2Fother%2Falgo_pdf.html)，刷题路线同时也开源在[Github](https://leetcode.cn/link/?target=https%3A%2F%2Fgithub.com%2Fyoungyangyang04%2Fleetcode-master)上，你会发现详见很晚！

如果感觉题解对你有帮助，不要吝啬给一个👍吧！

Sort默认从小到大排序，要从大到小排序可以用lambda函数

Sort(vec.begin(), vec.end(), [](int a, int b){ return a > b;});