完全背包

有N件物品和一个最多能背重量为W的背包。第i件物品的重量是weight[i]，得到的价值是value[i] 。**每件物品都有无限个（也就是可以放入背包多次），**求解将哪些物品装入背包里物品价值总和最大。

> 对于完全背包问题与01背包问题的区别在于:
> 完全背包问题中的**所有物品**都是有**无数个副本**,而01背包问题的**每个物品**只有一个**副本**。

在代码中的唯一不同就是

> 01背包和完全背包唯一不同就是体现在**遍历顺序上**。
> 非常有意思
> 我们知道**01背包内嵌**的循环是**从大到小遍历**，为了保证每个物品仅被添加一次
> 而**完全背包**的物品是可以添加多次的，所以要**从小到大去遍历**

```
//完全背包问题
for(int i=0;i<weight.size();i++){
 //内层循环
 for(int j=weight[i];j<=bagSize;j++){
 dp[j]=max(dp[j],dp[j-weight[i]]+value[i]);
 }
}
```

为什么遍历物品在外层循环，遍历背包容量在内层循环?

> 对于01背包来说**二维dp数组**的**两个for遍历**的**先后顺序**是可以颠倒的,一维dp数组的两个for循环先后循环一定是**先遍历物品**，再遍历**背包容量**

在完全背包中，对于一维dp数组来说，其实两个for循环嵌套顺序同样无所谓！

518. 零钱兑换 II

链接：https://leetcode-cn.com/problems/coin-change-2/

难度：中等

给定不同面额的硬币和一个总金额。写出函数来计算可以凑成总金额的硬币组合数。假设每一种面额的硬币有无限个。

示例 1:

输入: amount = 5, coins = [1, 2, 5]
输出: 4
解释: 有四种方式可以凑成总金额:
5=5
5=2+2+1
5=2+1+1+1
5=1+1+1+1+1

> **对于本文题问的是可以凑成总金额的硬币组合数，而不是能否凑成总金额；所以不能之前完全背包问题的套路**

而本题要求凑成总和的组合数，元素之间要求没有顺序。

所以纯完全背包是能凑成总结就行，不用管怎么凑的。

此时要改变的就是内外层循环的顺序了

此时内外层循环的先后顺序不同表达的含义不同

> 如果外层遍历**物品**,内层遍历**背包容量**,那么dp[j]里计算的是**组合数**
> 如果外层遍历**背包容量**,内层遍历**物品**,那么dp[j]里计算的是**排列数**

5步动态规划

> 1、确定dp数组及其下标(由于本题求得是组合数所以遍历顺序是先物品后容量)
> dp[j]:表示从下标0....i的硬币凑成总额为j的组合数
> 2、递推公式
> dp[j]+=dp[j-coins[i]];
> 3、初始化
> dp[0]=1;其余初始化0
> 4、遍历顺序:先硬币后凑成总额

```
class Solution {
public:
    int change(int amount, vector<int>& coins) {
 vector<int> dp(amount+1,0);
 dp[0]=1;
 //先物品后容量
 for(int i=0;i<coins.size();i++){
 //由于可以重复
 for(int j=coins[i];j<=amount;j++){
  dp[j]+=dp[j-coins[i]];
  }
  }
  return dp[amount];
      }
};
```

322. 零钱兑换

题目链接：https://leetcode-cn.com/problems/coin-change/

给定不同面额的硬币 coins 和一个总金额 amount。编写一个函数来计算可以凑成总金额所需的最少的硬币个数。如果没有任何一种硬币组合能组成总金额，返回 -1。

你可以认为每种硬币的数量是无限的。

示例 1：输入：coins = [1, 2, 5], amount = 11
输出：3
解释：11 = 5 + 5 + 1

示例 2：
输入：coins = [2], amount = 3
输出：-1

> 1、确定数组的下标及其定义

> dp[j]表示从下标为0....i的硬币,凑成j的硬币个数
> 2、递推公式
> dp[j]=min(dp[j],dp[j-coins[i]]+1):
> 3、初始化
> dp[0]=0;其余也初始化为0
> 4、遍历顺序:先硬币后金额，从金额小的遍历到金额大的

```
class Solution {
public:
    int coinChange(vector<int>& coins, int amount) {
 vector<int> dp(amount+1,1000000);
        dp[0]=0;
        for(int i=0;i<coins.size();i++){//先硬币后总金额
            for(int j=coins[i];j<=amount;j++){//硬币从小到大
                dp[j]=min(dp[j],dp[j-coins[i]]+1);
            }
         }
         return dp[amount]==1000000?-1:dp[amount];
     }
};
```