完全背包

知识

- **01背包**：内循环从后往前（去重）

  完全背包：内循环从前往后（复用）

- **组合**：外层物品i，内层背包j

  排列：外层背包j，内层物品i

模板

```
//（1）dp
vector<int> dp(bagSize+1, 0);
//（2）初始化
dp[0] = 0;          //min
dp[0] = 1;          //+=
dp[0] = INT_MAX;    //求最小
dp[0] = INT_MIN;    //求最大
//（3）核心遍历
//组合：外物品i，内背包j
//排列：外背包j，内物品i
for (int i=0; i<things.size(); i++) {           // 遍历物品
    //01背包：从后往前（去重）
    //完全背包：从前往后（复用）
    for (int j=things[i]; j<=bagSize; j++) {    // 遍历背包
        //更新dp
        //情况一：方法数
        dp[j] += dp[j - coins[i]];
        //情况二：排列（外背包j，内物品i）
        if(j-nums[i]>=0)    dp[j] += dp[j-nums[i]];
        //情况三：最少数（注意初始化）
        dp[j] = min(dp[j-things[i]]+1, dp[j]);
    }
}
//（4）return
return dp[bagSize];
```

518.零钱兑换II

```cpp
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];
    }
};
```

377.组合总和IV

```cpp
class Solution {
public:
    int combinationSum4(vector<int>& nums, int target) {
        vector<int> dp(target + 1, 0);
        dp[0] = 1;
        for (int i = 0; i <= target; i++) { // 遍历背包
            for (int j = 0; j < nums.size(); j++) { // 遍历物品
                if (i - nums[j] >= 0 && dp[i] < INT_MAX - dp[i - nums[j]]) {
                    dp[i] += dp[i - nums[j]];
                }
            }
        }
        return dp[target];
    }
};
```

70.爬楼梯

```cpp
class Solution {
public:
    int climbStairs(int n) {
        vector<int> dp(n + 1, 0);
        dp[0] = 1;
        for (int i = 1; i <= n; i++) { // 遍历背包
            for (int j = 1; j <= m; j++) { // 遍历物品
                if (i - j >= 0) dp[i] += dp[i - j];
            }
        }
        return dp[n];
    }
};
```

322.零钱兑换

```
class Solution {
public:
    int coinChange(vector<int>& coins, int amount) {
        vector<int> dp(amount + 1, INT_MAX);
        dp[0] = 0;
        for (int i = 0; i < coins.size(); i++) { // 遍历物品
            for (int j = coins[i]; j <= amount; j++) { // 遍历背包
                if (dp[j - coins[i]] != INT_MAX) { // 如果dp[j - coins[i]]是初始值
则跳过
                    dp[j] = min(dp[j - coins[i]] + 1, dp[j]);
                }
            }
        }
        if (dp[amount] == INT_MAX) return -1;
        return dp[amount];
    }
};
```

279.完全平方数

```
class Solution {
public:
    int numSquares(int n) {
        vector<int> dp(n + 1, INT_MAX);
        dp[0] = 0;
        for (int i = 1; i * i <= n; i++) { // 遍历物品
            for (int j = i * i; j <= n; j++) { // 遍历背包
                dp[j] = min(dp[j - i * i] + 1, dp[j]);
            }
        }
        return dp[n];
    }
};
```

139.单词拆分

```
class Solution {
public:
    bool wordBreak(string s, vector<string>& wordDict) {
        unordered_set<string> wordSet(wordDict.begin(), wordDict.end());
        vector<bool> dp(s.size() + 1, false);
        dp[0] = true;
        for (int i = 1; i <= s.size(); i++) {   // 遍历背包
            for (int j = 0; j < i; j++) {       // 遍历物品
                string word = s.substr(j, i - j); //substr(起始位置，截取的个数)
                if (wordSet.find(word) != wordSet.end() && dp[j]) {
                    dp[i] = true;
                }
            }
        }
        return dp[s.size()];
```

```
    }
};
```