Leetcode 总结版~动态规划

1. **最大子序和**

给定一个整数数组 nums ，找到一个具有最大和的连续子数组（子数组最少包含一个元素），返回其最大和。

示例:

输入: [-2,1,-3,4,-1,2,1,-5,4],

输出: 6

解释: 连续子数组 [4,-1,2,1] 的和最大，为 6。



1. **买卖股票的最佳时机**

给定一个数组，它的第 i 个元素是一支给定股票第 i 天的价格。

如果你最多只允许完成一笔交易（即买入和卖出一支股票），设计一个算法来计算你所能获取的最大利润。注意你不能在买入股票前卖出股票。

输入: [7,1,5,3,6,4]

输出: 5

解释: 在第 2 天（股票价格 = 1）的时候买入，在第 5 天（股票价格 = 6）的时候卖出，最大利润 = 6-1 = 5 。 注意利润不能是 7-1 = 6, 因为卖出价格需要大于买入价格。



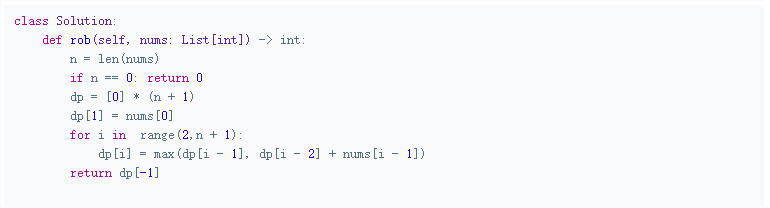
1. **打家劫舍**

你是一个专业的小偷，计划偷窃沿街的房屋。每间房内都藏有一定的现金，影响你偷窃的唯一制约因素就是相邻的房屋装有相互连通的防盗系统，如果两间相邻的房屋在同一晚上被小偷闯入，系统会自动报警。给定一个代表每个房屋存放金额的非负整数数组，计算你在不触动警报装置的情况下，能够偷窃到的最高金额。

输入: [2,7,9,3,1]

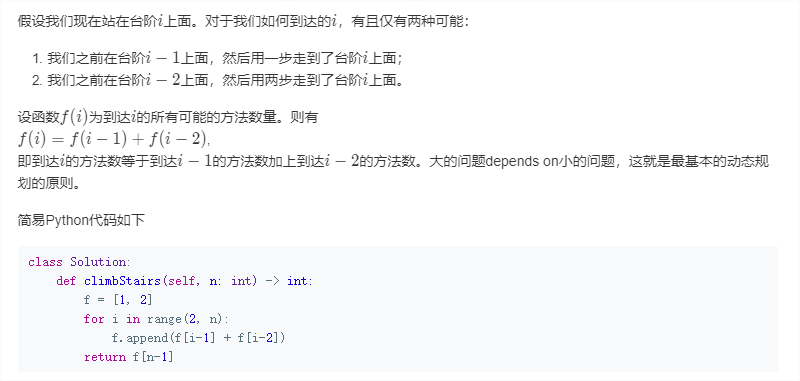
输出: 12

解释: 偷窃 1 号房屋 (金额 = 2), 偷窃 3 号房屋 (金额 = 9)，接着偷窃 5 号房屋 (金额 = 1)。偷窃到的最高金额 = 2 + 9 + 1 = 12 。



1. **爬楼梯**

假设你正在爬楼梯。需要 n 阶你才能到达楼顶。每次你可以爬 1 或 2 个台阶。你有多少种不同的方法可以爬到楼顶呢？



1. **零钱兑换**

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。

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. **买卖股票的最佳时机**



1. **单词拆分**

给定一个非空字符串 s 和一个包含非空单词列表的字典 wordDict，判定 s 是否可以被空格拆分为一个或多个在字典中出现的单词。

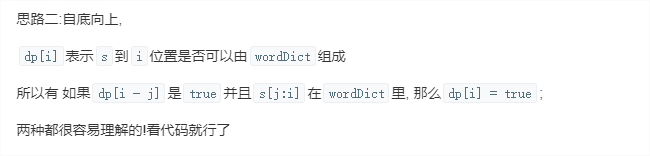
说明：拆分时可以重复使用字典中的单词。你可以假设字典中没有重复的单词。

输入: s = "applepenapple", wordDict = ["apple", "pen"]

输出: true

解释: 返回 true 因为 "applepenapple" 可以被拆分成 "apple pen apple"。

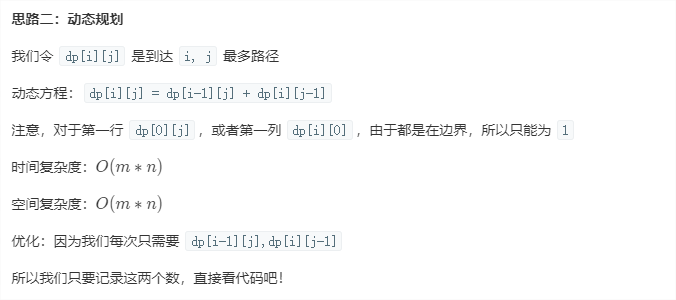
  注意你可以重复使用字典中的单词。

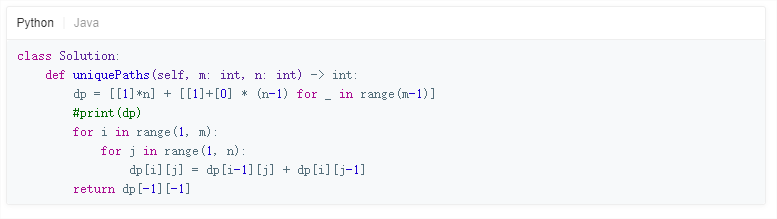




1. **机器人的运动路径**

一个机器人位于一个 m x n 网格的左上角 （起始点在下图中标记为“Start” ）。机器人每次只能向下或者向右移动一步。机器人试图达到网格的右下角（在下图中标记为“Finish”）。



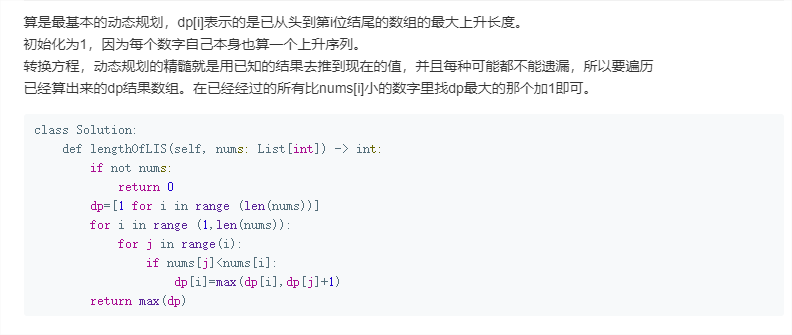


1. **最长上升子序列**

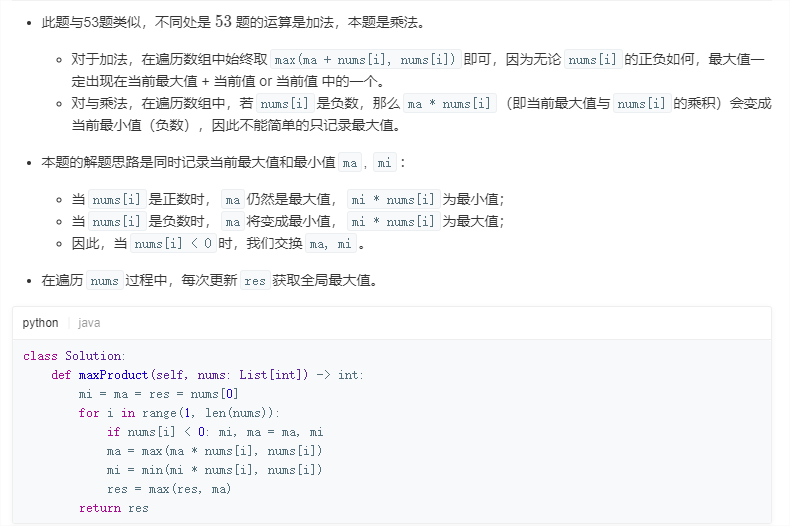
**输入:** [10,9,2,5,3,7,101,18]

**输出:** 4

**解释:** 最长的上升子序列是 [2,3,7,101]，它的长度是 4。



1. **乘积最大子序列**



1. **回文子串**

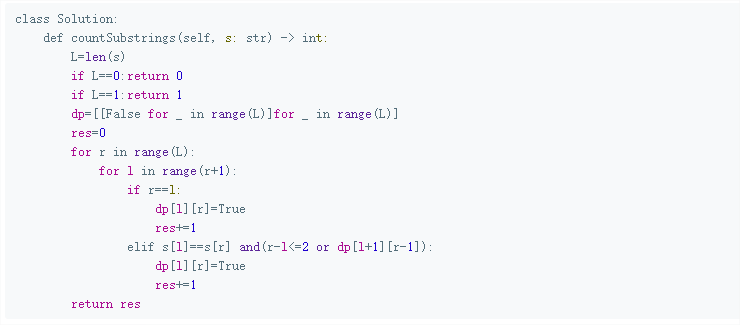
给定一个字符串，你的任务是计算这个字符串中有多少个回文子串。

具有不同开始位置或结束位置的子串，即使是由相同的字符组成，也会被计为是不同的子串。

输入: "abc"

输出: 3

解释: 三个回文子串: "a", "b", "c".



1. **最长的回文子串**



1. **目标和**

给定一个非负整数数组，a1, a2, ..., an, 和一个目标数，S。现在你有两个符号 + 和 -。对于数组中的任意一个整数，你都可以从 + 或 -中选择一个符号添加在前面。返回可以使最终数组和为目标数 S 的所有添加符号的方法数。

示例 1:

输入: nums: [1, 1, 1, 1, 1], S: 3

输出: 5

解释:

-1+1+1+1+1 = 3

+1-1+1+1+1 = 3

+1+1-1+1+1 = 3

+1+1+1-1+1 = 3

+1+1+1+1-1 = 3

一共有5种方法让最终目标和为3。

先使用数学思路，可以知道赋予标号后，集合中包含负数这正数，则有 sum(Positive)-sum(Negtive) = S。因为sum(Positive)+sum(Negtive)=sum(nums)，则有2\*sum(Positive)=sum(nums)+S，故sum(Positive)=(sum(nums)+S)/2，由于(sum(nums)+S)/2是固定的整数，所以只需要找到和为它的组合数即可。经过上述解析，可以将问题转化为从nums中找到和为(sum(nums)+S)/2的组合个数。这个问题可以通过动归来解决，用dp[i]存储和为i的组合数，然后对nums中的整数n进行遍历，对于所有i>=n的i，则有dp[i]=dp[i]+dp[i-n]

class Solution(object):

def findTargetSumWays(self, nums, S):

"""

:type nums: List[int]

:type S: int

:rtype: int

"""

if sum(nums)<S:

return 0

if (S+sum(nums))%2==1:

return 0

target = (S+sum(nums))/2

print target

dp = [0]\*(target+1)

dp[0] = 1

for n in nums:

i = target

while(i>=n):

dp[i] = dp[i] + dp[i-n]

i = i-1

return dp[target]

1. **分割等和子集**

给定一个**只包含正整数**的**非空**数组。是否可以将这个数组分割成两个子集，使得两个子集的元素和相等。

按照前一个的思路，把问题分割为dp[target]能不能实现的问题~然后有动态规划：

Dp[target] = 1 if dp[target-n] ==1 有点转化为twosum的感觉，最后自己的代码为：

class Solution(object):

def canPartition(self, nums):

"""

:type nums: List[int]

:rtype: bool

"""

if sum(nums) % 2 ==1:

return False

target = sum(nums) /2

dp = [0] \* (target+1)

dp[0] = 1

for n in nums:

i = target

while i>=n:

if dp[i-n] == 1:

dp[i] = 1

i-=1

return dp[target] == 1