本日学习内容

1. 继续完成管理系统
2. dp的深化，随想录写到16
3. 二叉树部分数据结构

本日分享内容

题目1.

给你整数 `zero` ， `one` ， `low` 和 `high` ，我们从空字符串开始构造一个字符串，每一步执行下面操作中的一种：

- 将 `'0'` 在字符串末尾添加 `zero` 次。
- 将 `'1'` 在字符串末尾添加 `one` 次。

以上操作可以执行任意次。

如果通过以上过程得到一个 **长度** 在 `low` 和 `high` 之间（包含上下边界）的字符串，那么这个字符串我们称为 **好** 字符串。

请你返回满足以上要求的 **不同** 好字符串数目。由于答案可能很大，请将结果对 `109 + 7` **取余** 后返回。

示例 1：

```
输入：low = 3, high = 3, zero = 1, one = 1
输出：8
解释：
一个可能的好字符串是 "011" 。
可以这样构造得到："" -> "0" -> "01" -> "011" 。
从 "000" 到 "111" 之间所有的二进制字符串都是好字符串。
```

示例 2：

```
输入：low = 2, high = 3, zero = 1, one = 2
输出：5
解释：好字符串为 "00" ，"11" ，"000" ，"110" 和 "011" 。
```

提示：

- `1 <= low <= high <= 105`
- `1 <= zero, one <= low`

思路

dp[i] 表示：构造长度为 i 的字符串的方案数（即有多少种构造方式可以得到长度为 i 的“好字符串”）

如果我已经能构造出长度为 i - zero 的字符串，加上一个 zero 长度的全 0 子串 → 长度变成 i

同理，加上 one 长度的全 1 子串 → 也是构造长度为 i 的一种方式

由上得出状态转移方程，写出代码

注意初始值为1

代码

```
const int MODULO = 1000000007;

class Solution {
public:
    int countGoodStrings(int low, int high, int zero, int one) {
        vector<int> dp(high + 1);
        dp[0] = 1;
        for (int i = 1; i <= high; i++) {
            if (i >= zero) {
                dp[i] = (dp[i] + dp[i - zero]) % MODULO;
            }
            if (i >= one) {
                dp[i] = (dp[i] + dp[i - one]) % MODULO;
            }
        }
        int total = 0;
        for (int i = low; i <= high; i++) {
            total = (total + dp[i]) % MODULO;
        }
        return total;
    }
};
```

题目2.

给你一个由 **不同** 整数组成的数组 `nums` ，和一个目标整数 `target` 。请你从 `nums` 中找出并返回总和为 `target` 的元素组合的个数。

题目数据保证答案符合 32 位整数范围。

示例 1：

```
输入：nums = [1,2,3], target = 4
输出：7
解释：
所有可能的组合为：
(1, 1, 1, 1)
(1, 1, 2)
(1, 2, 1)
(1, 3)
(2, 1, 1)
(2, 2)
(3, 1)
请注意，顺序不同的序列被视作不同的组合。
```

示例 2：

```
输入：nums = [9], target = 3
输出：0
```

提示：

- `1 <= nums.length <= 200`
- `1 <= nums[i] <= 1000`
- `nums` 中的所有元素 **互不相同**
- `1 <= target <= 1000`

思路

dp[i]表示和为i的排列数，对于每个i，遍历nums中的数字num，如果 i >= num 则 dp[i] += dp[i - m];

需注意使用unsigned int。题中也特别提出保证答案符合32位整数范围

代码

</> 代码

C++ 智能模式

```cpp
class Solution {
public:
    int combinationSum4(vector<int>& nums, int target) {
        int n = nums.size();
        vector<unsigned int> dp(target + 1, 0);
        dp[0] = 1;

        for (int i = 1; i <= target; i++) {
            for (int num : nums) {
                if (i >= num) {
                    dp[i] += dp[i - num];
                }
            }
        }
        return dp[target];
    }
};
```

本日遇到的问题

1. 依旧不能在背包问题的各种变式中准确识别出背包模型
2. 有些题目不能推导出递归过程，导致理解不够透彻，还需要继续深入学习理解。

明日学习内容

1. 继续dp部分的学习，写到随想录20
2. Leetcode每日一题
3. 完善管理系统
4. 抽时间继续看大话数据结构