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20，完成部分灵神题单
3. 复习二分，开始使用upper_bound, lower_bound 简化解题过程

本日分享内容

题目1.

给你一个非负整数数组 `nums` 和一个整数 `target` 。

向数组中的每个整数前添加 `'+'` 或 `'-'` ，然后串联起所有整数，可以构造一个 **表达式**：

- 例如，`nums = [2, 1]` ，可以在 `2` 之前添加 `'+'` ，在 `1` 之前添加 `'-'` ，然后串联起来得到表达式 `"+2-1"` 。

返回可以通过上述方法构造的、运算结果等于 `target` 的不同 **表达式** 的数目。

示例 1：

```
输入：nums = [1,1,1,1,1], target = 3
输出：5
解释：一共有 5 种方法让最终目标和为 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
+1 + 1 + 1 + 1 - 1 = 3
```

示例 2：

```
输入：nums = [1], target = 1
输出：1
```

思路

left + right = sum，而sum是固定的。left - (sum - left) = target 推导出 left = (target + sum)/2 。

target是固定的，sum是固定的，left就可以求出来。此时问题就是在集合nums中找出和为left的组合。

代码

```
public:
    int findTargetSumWays(vector<int>& nums, int target) {
        int sum = 0;
        for (int i = 0; i < nums.size(); i++) sum += nums[i];
        if (abs(target) > sum) {
            return 0; // 此时没有方案
        }
        if ((target + sum) % 2 == 1) {
            return 0; // 此时没有方案
        }
        int bagSize = (target + sum) / 2;
        vector<int> dp(bagSize + 1, 0);
        dp[0] = 1;
        for (int i = 0; i < nums.size(); i++) {
            for (int j = bagSize; j >= nums[i]; j--) {
                dp[j] += dp[j - nums[i]];
            }
        }
        return dp[bagSize];
    }
```

题目2.

- 给你一个二进制字符串数组 `strs` 和两个整数 `m` 和 `n` 。

  请你找出并返回 `strs` 的最大子集的长度，该子集中 **最多** 有 `m` 个 `0` 和 `n` 个 `1` 。

  如果 `x` 的所有元素也是 `y` 的元素，集合 `x` 是集合 `y` 的 **子集** 。

  示例 1：

  ```
  输入：strs = ["10", "0001", "111001", "1", "0"], m = 5, n = 3
  输出：4
  解释：最多有 5 个 0 和 3 个 1 的最大子集是 {"10","0001","1","0"} ，因此答案是 4 。
  其他满足题意但较小的子集包括 {"0001","1"} 和 {"10","1","0"} 。{"111001"} 不满足题意，因为它含 4 个 1 ，大于 n 的值 3 。
  ```

  示例 2：

  ```
  输入：strs = ["10", "0", "1"], m = 1, n = 1
  输出：2
  解释：最大的子集是 {"0", "1"} ，所以答案是 2 。
  ```

提示：

- `1 <= strs.length <= 600`
- `1 <= strs[i].length <= 100`
- `strs[i]` 仅由 `'0'` 和 `'1'` 组成
- `1 <= m, n <= 100`

思路

每个字符串相当于一个物品，`'0'` 和 `'1'` 的数量相当于物品的两种“重量”，背包容量为 `(m, n)`。`dp[i][j]` 表示使用 `i` 个 `'0'` 和 `j` 个 `'1'` 时能组成的最大字符串数量，对于每个字符串 `s`，统计其 `'0'` 的数量 `zeros` 和 `'1'` 的数量 `ones`。

代码

```cpp
class Solution {
public:
    int findMaxForm(vector<string>& strs, int m, int n) {
        vector<vector<int>> dp(m + 1, vector<int>(n + 1, 0));

        for (const string& s : strs) {
            int zeros = count(s.begin(), s.end(), '0');
            int ones = s.size() - zeros;

            for (int i = m; i >= zeros; --i) {
                for (int j = n; j >= ones; --j) {
                    dp[i][j] = max(dp[i][j], dp[i - zeros][j - ones] + 1);
                }
            }
        }

        return dp[m][n];
    }
};
```

本日遇到的问题

1. dp进度缓慢，部分题目个人感觉难度较高，有些难以推进
2. 二分的部分问题中，if判断中的取等条件（是否加‘=’）需要再深入思考总结

明日学习内容

1. 重温灵神二分题单，复习并解决历史遗留问题

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