基于阿珍对01背包的解释的拓展

阿珍说：01背包是可以解决：一组数字中，可以找到两个相加起来最相近的集合，集合的数字的总数就是数组总数

她只是看了具体的题型

416. 分割等和子集

力扣题目链接

题目难易：中等

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。

注意: 每个数组中的元素不会超过 100 数组的大小不会超过 200

示例 1: 输入: [1, 5, 11, 5] 输出: true 解释: 数组可以分割成 [1, 5, 5] 和 [11].

示例 2: 输入: [1, 2, 3, 5] 输出: false 解释: 数组不能分割成两个元素和相等的子集.

提示:

- 1 <= nums.length <= 200
- 1 <= nums[i] <= 100

只是大体的了解了这道题目

实质我们进一步理解

就是往一个规定数值容器中放数（这个数在一个数组中，数组中的数都可以拿，这里只能拿一次），一定要达到容器的总和，达不到就死吧。

for（。。。。）//这里就直接遍历放的数

for（。。。。）//这里直接遍历我的容器 （由低到高就是可以重复拿，由高到底不可重复拿），这里不可以重复拿如上面黄色标志可知，不可重复拿。

那么这里加粗的才是我们需要理解的地方，dp关系我们一眼可知 很简单 是我们尽全力满足容器！

```
dp[j] = Math.max(dp[j], dp[j-nums[i]] + nums[i]);
```

1049. 最后一块石头的重量 II

力扣题目链接

题目难度：中等

有一堆石头，每块石头的重量都是正整数。

每一回合，从中选出任意两块石头，然后将它们一起粉碎。假设石头的重量分别为 x 和 y，且 x <= y。那么粉碎的可能结果如下：

如果 x == y，那么两块石头都会被完全粉碎；如果 x != y，那么重量为 x 的石头将会完全粉碎，而重量为 y 的石头新重量为 y-x。最后，最多只会剩下一块石头。返回此石头最小的可能重量。如果没有石头剩下，就返回 0。

示例：输入：[2,7,4,1,8,1] 输出：1 解释：组合 2 和 4，得到 2，所以数组转化为 [2,7,1,8,1]，组合 7 和 8，得到 1，所以数组转化为 [2,1,1,1]，组合 2 和 1，得到 1，所以数组转化为 [1,1,1]，组合 1 和 1，得到 0，所以数组转化为 [1]，这就是最优值。

提示：

- 1 <= stones.length <= 30
- 1 <= stones[i] <= 1000

这一题和上面一题一样 不赘述 easy

494. 目标和

力扣题目链接

难度：中等

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：

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。

提示：

- 数组非空，且长度不会超过 20 。
- 初始的数组的和不会超过 1000 。
- 保证返回的最终结果能被 32 位整数存下。

又来一题 由感而发 区别在于此 dp关系式

```
dp[j] += dp[j - nums[i]]//方案数的dp关系式
```

474.一和零

力扣题目链接

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 的最大子集的大小，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

首先拿到题目 转化题意：为了尽可能不超过5 3的背包容量我们最多可以返回最大子集的长度，而不是问你我们尽可能不超过5 3的背包容量我们最多可以返回最大子集的具体值，所以我们立刻想到

```
dp[j] += dp[j - nums[i]]//方案数的dp关系式
```

而不是

```
dp[j] = Math.max(dp[j], dp[j-nums[i]] + nums[i]);
```

并且这里只可以拿一个，不可重复多拿 所以

for先字符串 指代物品

for 5 3 指代容量

这里是两个所以就两个维度