给定一个整数数组和一个目标值，找出数组中和为目标值的两个数。

你可以假设每个输入只对应一种答案，且同样的元素不能被重复利用。

（数组里的数不重复：即对同一个数字以及和数而言只可能有一种数字搭配。）

示例:

给定 nums = [2, 7, 11, 15], target = 9

因为 nums[0] + nums[1] = 2 + 7 = 9

所以返回 [0, 1]

===========================================================

class Solution {

public:

vector<int> twoSum(vector<int>& nums, int target) {

vector<int> sum;

for (int i = 0; i < nums.size() - 1; ++i)

{

// 这样会导致重复判断，不可行

if (nums[i] + nums[i + 1] == target)

{

sum.push\_back(i);

sum.push\_back(i + 1);

++i;

}

}

return sum;

}

};

7 / 20 个通过测试用例

状态：解答错误

输入：

[3,2,3]

6

输出：

[]

预期：

[0,2]

===========================================================

class Solution {

public:

vector<int> twoSum(vector<int>& nums, int target) {

vector<int> sum;

for (int i = 0; i < nums.size(); ++i)

{

for (int j = i + 1; j < nums.size(); ++j)

{

if (nums[i] + nums[j] == target)

{

sum.push\_back(i);

sum.push\_back(j);

nums[i] = nums[j] = NULL; // “同样的元素不能被重复利用”？

}

}

}

return sum;

}

};

17 / 20 个通过测试用例

状态：解答错误

输入：

[572,815,387,418,434,530,376,190,196,74,830,561,973,771,640,37,539,369,327,51,623,575,988,44,659,48,22,776,487,873,486,169,499,82,128,31,386,691,553,848,968,874,692,404,463,285,745,631,304,271,40,921,733,56,883,517,99,580,55,81,232,971,561,683,806,994,823,219,315,564,997,976,158,208,851,206,101,989,542,985,940,116,153,47,806,944,337,903,712,138,236,777,630,912,22,140,525,270,997,763,812,597,806,423,869,926,344,494,858,519,389,627,517,964,74,432,730,843,673,985,819,397,607,34,948,648,43,212,950,235,995,76,439,614,203,313,180,760,210,813,920,229,615,730,359,863,678,43,293,978,305,106,797,769,3,700,945,135,430,965,762,479,152,121,935,809,101,271,428,608,8,983,758,662,755,190,632,792,789,174,869,622,885,626,310,128,233,82,223,339,771,741,227,131,85,51,361,343,641,568,922,145,256,177,329,959,991,293,850,858,76,291,134,254,956,971,718,391,336,899,206,642,254,851,274,239,538,418,21,232,706,275,615,568,714,234,567,994,368,54,744,498,380,594,415,286,260,582,522,795,261,437,292,887,405,293,946,678,686,682,501,238,245,380,218,591,722,519,770,359,340,215,151,368,356,795,91,250,413,970,37,941,356,648,594,513,484,364,484,909,292,501,59,982,686,827,461,60,557,178,952,218,634,785,251,290,156,300,711,322,570,820,191,755,429,950,18,917,905,905,126,790,638,94,857,235,889,611,605,203,859,749,874,530,727,764,197,537,951,919,24,341,334,505,796,619,492,295,380,128,533,600,160,51,249,5,837,905,747,505,82,158,687,507,339,575,206,28,29,91,459,118,284,995,544,3,154,89,840,364,682,700,143,173,216,290,733,525,399,574,693,500,189,590,529,972,378,299,461,866,326,43,711,460,426,947,391,536,26,579,304,852,158,621,683,901,237,22,225,59,52,798,262,754,649,504,861,472,480,570,347,891,956,347,31,784,581,668,127,628,962,698,191,313,714,893]

101

输出：

[83,239,166,239]

预期：

[83,239]

nums[83] = 47; nums[239] = 54; nums[166] = 101; // 置为NULL后导致出错

===========================================================

class Solution {

public:

vector<int> twoSum(vector<int>& nums, int target) {

vector<int> sum;

for (int i = 0; i < nums.size(); ++i)

{

for (int j = i + 1; j < nums.size(); ++j)

{

if (nums[i] + nums[j] == target)

{

sum.push\_back(i);

sum.push\_back(j);

//nums[i] = nums[j] = NULL; // [2,3,3,3]

}

}

}

return sum;

}

};

20 / 20 个通过测试用例

状态：通过

执行用时：204 ms

~~我的输入~~

~~[2, 3, 3, 3]~~

~~6~~

~~我的答案~~

~~[1,2,1,3,2,3]~~

~~预期答案~~

~~[1,3]~~

===========================================================

class Solution {

public:

vector<int> twoSum(vector<int>& nums, int target) {

vector<int> sum;

for (int i = 0; i < nums.size(); ++i)

{

for (int j = i + 1; j < nums.size(); ++j)

{

if (nums[i] + nums[j] == target && nums[i] != NULL && nums[j] != NULL)

{

sum.push\_back(i);

sum.push\_back(j);

nums[i] = nums[j] = NULL;

}

}

}

return sum;

}

};

19 / 20 个通过测试用例

状态：解答错误

输入：

[0,4,3,0]

0

输出：

[]

预期：

[0,3]

NULL就是0，是个宏

#define NULL 0

<https://www.zhihu.com/question/22203461> C/C++ 中 0 与 NULL 区别是什么？

===========================================================

给一整数数组, 找到数组中有多少组 不同的元素对 有相同的和, 且和为给出的 target 值, 返回对数.

class Solution {

public:

/\*

\* @param nums: an array of integer

\* @param target: An integer

\* @return: An integer

\*/

int twoSum6(vector<int> &nums, int target) {

// write your code here

if(nums.size() < 2) return 0;

int l = 0, r = nums.size()-1;

int count = 0;

sort(nums.begin(), nums.end());

while(l < r) {

if(l > 0 && nums[l] == nums[l-1]) {

l++;

continue;

}

if(nums[l] + nums[r] == target) {

count++;

l++;

r--;

} else if (nums[l] + nums[r] < target) {

l++;

} else {

r--;

}

}

return count;

}

};

双指针法。

与java版本的思想一样。去重的步骤稍微不同，在本题的要求下，左右两指针对应的元素和目标是一定的，只要控制一端不重复即可。

所以在while循环中，先进行判断是否left与前一个元素相同，如是则left++并continue。