1、Two Sum

Q：

Given an array of integers, return indices of the two numbers such that they add up to a specific target.

You may assume that each input would have *exactly* one solution.

Example:

Given nums = [2, 7, 11, 15], target = 9,

Because nums[0] + nums[1] = 2 + 7 = 9,

return [0, 1].

A：

解决方案一：

扫描数组，存放到HashMap中<arr[i],i>。再次扫描数组，在map中寻找另一个加数，不存在，就pass，存在，若两个加数不相等，返回；两个加数相等，就再扫描整个数组，看是否存在两个加数，若是，返回；不是，pass掉。

时间复杂度分析：

时间复杂度取决于是否另外一个加数的下标是否等于2，平均时间复杂度O（n），最坏时间复杂度是n^2，但是实践复杂度接近O（n）.

JAVA代码：

import java.util.\*;

public class Solution {

/\*\*

\*返回值：一个数组引用，数组存储的是两个符合要求的数组元素下标

\*参数列表:数组引用，目的和

\*/

public static int[] twoSum(int[] arr,int target) {

Map<Integer,Integer> map = new HashMap<>();

//遍历数组，将<数组元素，元素下标>存储在Map中，

//Map中，重复的数组元素只保留第一次扫描到的。

for(int i = 0;i < arr.length;++i)

map.put(arr[i],i);

for(int i = 0;i < arr.length;++i) {

int another = target - arr[i];

/\*数组含有和为sum的元素\*/

if(map.containsKey(arr[i]) && map.get(another) != null) {

//两个元素相等时，需要排除是同一个元素

if(map.get(another)== i) {

for(int j = 0;j < arr.length;++j) {

if(arr[j] == another && j != i) {

int[] res = {i,j};

return res;

}

}

}

else {//两个元素不相等

int[] res = {i,map.get(another)};

return res;

}

}

}

return null;

}

}

解决方法二

1. 对数组排序，时间复杂度是nlogn；
2. 再对有序数组操作，时间复杂度是n，总时间复杂度是nlogn. 头尾相加大于sum，排除尾；小于sum，排除头。

这里给出(2)的C代码：

int TwoSum(int\* arr,int len,int sum,int\* one,int\* another) {

int i = 0;

int j = len - 1;

while(i < j) {

if(arr[i]+arr[j] == sum) {

\*one = i;

\*another = j;

return 1;

}

else if(arr[i]+arr[j] < sum) {

++i;

}

else {

--j;

}

}

return 0;

}