##Problem Description

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, and you may not use the same element twice.

\*\*Example\*\*

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

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

>return [0, 1].

##The Thinking

\* Assumptions

\* The is exactly one solution.

\* A number can only appear more than once in the array.

\* We can't use the same element twice.

\* Algorithm

\* Hash table. The first thing we should be aware of is that the target number is the result of the sum of two different elements. So, if we scan the array, and for each element, we check whether or not `target-current\_element` exists. If `target-current\_element` exists, the solution is found.

In order to implement this solution, we need to build a hash table which map number with the index in the array. If a number doesn't appear in the array, we make the value -1.

Also we should notice that the number can be negative, may be it is a better way for us to use relative index. I mean we put the smallest number index 0, and assign the difference between number and the smallest number as the index for current number. Moreover, pay attention integer overflow.

\* Binary Search. First we sort the copy of array`nums`,i.e. `tmp` in the increasing order. Then use two pointer, `head` and `rear`, which are initialized to `0` and `len-1`. Since this array is sorted, we can find the solution by altering head and rear.

If `tmp[head]+tmp[rear] > target`, then rear should be smaller to satisfy the equality. And if `tmp[head]+tmp[rear] < target`, head should be larger.

The time when `tmp[head]+tmp[rear] = target`, head and rear are the indices we want.

##Solution

\* Hash Table Solution

vector<int> twosum(vector<int> &nums, int target)

{

vector<int> res(2,0);

int len = nums.size();

int maxone = nums[0];

int minone = nums[0];

for (int i = 1; i < len; ++i){

maxone = max(maxone, nums[i]);

minone = min(minone, nums[i]);

}

maxone = maxone - minone;

//int arr[20000]={-1};

for (int i = 0; i < 20000; ++i){

arr[i] = -1;

}

for (int i = 0; i < len; ++i){

arr[nums[i]-minone] = i;

}

for (int i = 0; i < len; ++i){

if (target-nums[i]-minone >= 0 && arr[target-nums[i]-minone] > -1 && arr[target-nums[i]-minone] != i){

res[0] = i;

res[1] = arr[target-nums[i]-minone];

break;

}

}

return res;

}

\* Binary Search Solution

vector<int> res(2,0);

int len = nums.size();

vector<int> tem = nums;

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

int head = 0;

int rear = len - 1;

int tmp;

while ((tmp = nums[head]+nums[rear]) != target)

{

if (tmp > target){

rear--;

}

else {

head++;

}

}

vector<int>::iterator first = find(tem.begin(), tem.end(), nums[head]);

vector<int>::iterator second = find(tem.begin(), tem.end(), nums[rear]);

if (second==first){

second = find(first+1, tem.end(), nums[rear]);

}

int index1 = distance(tem.begin(), first);

int index2 = distance(tem.begin(), second);

res[0] = min(index1, index2);

res[1] = max(index1, index2);

return res;