A non-empty zero-indexed array A consisting of N positive integers is given. A pair of indices (P, Q), such that 0 ≤ P ≤ Q < N, is called a slice of array A. The sum of a slice (P, Q) is the total of A[P] + A[P+1] + ... + A[Q].

The equi-3 pair is a pair of indices (X, Y), such that 0 < X, X + 1 < Y < N − 1, and the sums of slices (0, X−1), (X+1, Y−1), (Y+1, N−1) are all equal.

For example, consider array A such that:

A[0] = 4

A[1] = 5

A[2] = 1

A[3] = 1

A[4] = 1

A[5] = 1

A[6] = 4

A[7] = 3

A[8] = 1

Pair (1, 6) is an equi-3 pair, because the sums of slices (0, 0), (2, 5) and (7, 8) are all equal to 4. Pair (2, 4) is not an equi-3 pair, because although the sums of slices (0, 1) and (5, 8) are both equal to 9, the sum of the middle slice (3, 3) equals 1. There is only one equi-3 pair in this array.

The goal is to check whether array A contains an equi-3 pair. Write a function:

class Solution { public int solution(int[] A); }

that, given a non-empty zero-indexed array A consisting of N integers, returns 1 if array A contains an equi-3 pair and 0 otherwise.

For example, given array A shown above, the function should return 1, as explained above.

Assume that:

* N is an integer within the range [5..100,000];
* each element of array A is an integer within the range [1..10,000].

Complexity:

* expected worst-case time complexity is O(N);
* expected worst-case space complexity is O(N), beyond input storage (not counting the storage required for input arguments).

Elements of input arrays can be modified.

由题可得，即使数组有10000个元素，每个均为最大数100000, 数组和也小于2147483647，用int就可以表示。

思路：

1. 判断给定数组是否符合题意并且算出数组和
2. 定义first和last作为Pair成员
3. 两个成员按照规则向数组中间靠拢并判断数组是否符合题意
4. 最终时间复杂度为O(2n)即为O(n) 空间复杂度为O(7)即为O(1)

Code C++

#include <iostream>

#include <vector>

using namespace std;

int isEqui\_3Array(vector<int> arr){

int len = arr.size();

if(len<5 || len>100000)

return 0;

int sum = 0;//sum of array

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

if(arr[i] <= 0)

return 0;

else

sum += arr[i];

}

int first = 1;//Pair 0

int last = len-2;//Pair 1

int sumOfFirst = arr[first-1];

int sumOfLast = arr[last+1];

int sumOfMin = sum - (sumOfFirst+arr[first]+arr[last]+sumOfLast);

while(first < last){

if(sumOfFirst==sumOfMin && sumOfMin==sumOfLast)

return 1;

if(sumOfLast<sumOfFirst && sumOfLast<sumOfMin){//sumOfLast<sumOfFirst<sumOfMin

sumOfLast += arr[last];

--last;

sumOfMin -= arr[last];

continue;

}else if(sumOfFirst<sumOfLast && sumOfLast<sumOfMin){//sumOfFirst<sumOfLast<sumOfMin

sumOfFirst += arr[first];

++first;

sumOfMin -= arr[first];

continue;

}else if(sumOfFirst==sumOfLast && sumOfLast<sumOfMin){//sumOfFirst==sumOfLast<sumOfMin

sumOfFirst += arr[first];

sumOfLast += arr[last];

--last;

++first;

sumOfMin -= (arr[first]+arr[last]);

continue;

}else{

return 0;

}

}

return 0;

}

int main()

{

//鲁棒测试

vector<int> test0 = {4,5,1,1,1,1,4,3,1};//true, return 1

vector<int> test1 = {5,5,5,2,5,5,5,2,5,5,5};//true, return 1

vector<int> test2 = {5,5,5,6,5,5,5,9,5,5};//false, return 0

vector<int> test3 = {5,5,6,5,5,5,9,5,5,5};//false, return 0

vector<int> test4 = {5,5,9,5,5,5,9,5,5}; //false, return 0

vector<int> test5 = {1,1,1,-2,1,1,1,2,1,1,1};// non-zero-indexed array false, return 0

vector<int> test6 = {};// empty array false, return 0

vector<int> test7 = {1,3,5,1,1,1,1,4,4};//true, return 1

vector<int> test8 = {1,1,1,1,9,1,1,1,1};//false, return 0

cout << isEqui\_3Array(test0) << endl;

cout << isEqui\_3Array(test1) << endl;

cout << isEqui\_3Array(test2) << endl;

cout << isEqui\_3Array(test3) << endl;

cout << isEqui\_3Array(test4) << endl;

cout << isEqui\_3Array(test5) << endl;

cout << isEqui\_3Array(test6) << endl;

cout << isEqui\_3Array(test7) << endl;

cout << isEqui\_3Array(test8) << endl;

return 0;

}