4、Median of Two Sorted Arrays

There are two sorted arrays **nums1** and **nums2** of size m and n respectively. Find the median of the two sorted arrays. The overall run time complexity should be O(log (m+n)).

思路一：

直接做融合、排序，再找中位数。

使用algorithm中的sort函数，采用快速排序算法，时间复杂度nlogn。（80ms）

double Solution**::**findMedianSortedArrays**(**vector**<**int**>&** nums1**,** vector**<**int**>&** nums2**)**

**{**

int size1 **=** nums1**.**size**(),** size2 **=** nums2**.**size**();**

**if** **(**size1 **==** 0**)** **return** findMedian**(**nums2**);**

**if** **(**size2 **==** 0**)** **return** findMedian**(**nums1**);**

**if** **(**size1 **<** size2**)** **return** findMedianSortedArrays**(**nums2**,** nums1**);**

// if (size1 == size2) return (findMedian(nums1) + findMedian(nums2)) / 2;

**for** **(**int i **=** 0**;** i **<** size2**;** i**++)**

**{**

nums1**.**push\_back**(**nums2**[**i**]);**

**}**

std**::**sort**(**nums1**.**begin**(),** nums1**.**end**());**

**return** findMedian**(**nums1**);**

**}**

double Solution**::**findMedian**(**vector**<**int**>&** nums**)**

**{**

**if** **(**nums**.**size**()** **%** 2**)**

**return** nums**[**nums**.**size**()** **/** 2**];**

**else**

**return** **((**double**)**nums**[**nums**.**size**()** **/** 2**]** **+** nums**[**nums**.**size**()** **/** 2 **-** 1**])** **/** 2**;**

**}**

思路二：

将问题转换为寻找两个数组第k小数值的问题，本题中，需要分奇偶情况进行处理。时间复杂度lgn。

class Solution **{**

public**:**

double findMedianSortedArrays**(**vector**<**int**>&** nums1**,** vector**<**int**>&** nums2**)** **{**

int m **=** nums1**.**size**(),** n **=** nums2**.**size**();**

int A**[**1000**],** B**[**1000**];**

**for** **(**int i **=** 0**;** i **<** m**;** i**++)**

**{**

A**[**i**]** **=** nums1**[**i**];**

**}**

**for** **(**int i **=** 0**;** i **<** n**;** i**++)**

**{**

B**[**i**]** **=** nums2**[**i**];**

**}**

int total **=** m **+** n**;**

**if** **(**total **&** 0x1**)**

**return** findKth**(**A**,** m**,** B**,** n**,** total **/** 2 **+** 1**);**

**else**

**return** **(**findKth**(**A**,** m**,** B**,** n**,** total **/** 2**)**

**+** findKth**(**A**,** m**,** B**,** n**,** total **/** 2 **+** 1**))** **/** 2**;**

**}**

double findKth**(**int a**[],** int m**,** int b**[],** int n**,** int k**)**

**{**

//always assume that m is equal or smaller than n

**if** **(**m **>** n**)**

**return** findKth**(**b**,** n**,** a**,** m**,** k**);**

**if** **(**m **==** 0**)**

**return** b**[**k **-** 1**];**

**if** **(**k **==** 1**)**

**return** min**(**a**[**0**],** b**[**0**]);**

//divide k into two parts

int pa **=** min**(**k **/** 2**,** m**),** pb **=** k **-** pa**;**

**if** **(**a**[**pa **-** 1**]** **<** b**[**pb **-** 1**])**

**return** findKth**(**a **+** pa**,** m **-** pa**,** b**,** n**,** k **-** pa**);**

**else** **if** **(**a**[**pa **-** 1**]** **>** b**[**pb **-** 1**])**

**return** findKth**(**a**,** m**,** b **+** pb**,** n **-** pb**,** k **-** pb**);**

**else**

**return** a**[**pa **-** 1**];**

**}**

**};**

### 7、Reverse Integer

Reverse digits of an integer.

**Example1:** x = 123, return 321  
**Example2:** x = -123, return -321

**Have you thought about this?**

Here are some good questions to ask before coding. Bonus points for you if you have already thought through this!

If the integer's last digit is 0, what should the output be? ie, cases such as 10, 100.

Did you notice that the reversed integer might overflow? Assume the input is a 32-bit integer, then the reverse of 1000000003 overflows. How should you handle such cases?

For the purpose of this problem, assume that your function returns 0 when the reversed integer overflows.

思路：从个位开始扫描一遍数据，边扫描边组建。注意区分正负数的情况，溢出判断有所区别。

(时间复杂度O（n），n是数据的位数)

class Solution **{**

public**:**

int reverse**(**int x**)** **{**

int quotient **=** x**,** remainder **=** 0**;**

int res **=** 0**;**

int flag **=** 0**;**

**while** **(**quotient**)**

**{**

remainder **=** quotient **%** 10**;**

quotient **=** quotient **/** 10**;**

**if** **(**remainder **==** 0 **&&** flag **==** 0**)** **continue;**

flag **=** 1**;**

**if** **(**x **>** 0**){**

**if** **((**INT\_MAX **-** remainder**)** **/** 10 **<** res**)**

**{**

res **=** 0**;**

**break;**

**}**

res **=** res **\*** 10 **+** remainder**;**

**}**

**else**

**{**

**if** **((**INT\_MIN **-** remainder**)** **/** 10 **>** res**)**

**{**

res **=** 0**;**

**break;**

**}**

res **=** res **\*** 10 **+** remainder**;**

**}**

**}**

**return** res**;**

**}**

**};**

### 9、Palindrome Number

Determine whether an integer is a palindrome. Do this without extra space.

[click to show spoilers.](https://leetcode.com/problems/palindrome-number/)

**Some hints:**

Could negative integers be palindromes? (ie, -1)

If you are thinking of converting the integer to string, note the restriction of using extra space.

You could also try reversing an integer. However, if you have solved the problem "Reverse Integer", you know that the reversed integer might overflow. How would you handle such case?

There is a more generic way of solving this problem.

思路一：

Reverse Integer，因为全是正数，只考虑溢出的情况，可以用一个longlong型的整数存反转的数字，然后比较是否相等。

思路二：

首先计算数据的位数，然后从两端向中间扫描数据，判断每位是否相等。（80ms）

Ps：题中要求不能使用额外空间，所以排除了使用字符串、栈等可能。

class Solution **{**

public**:**

bool isPalindrome**(**int x**)** **{**

**if** **(**x **<** 0**)** **return** **false;**

**if** **(**x **==** 0**)** **return** **true;**

int quotient **=** x**;**

int frontdiv **=** 1**,** reardiv **=** 1**;**

**while** **(**quotient**/**10**)**

**{**

frontdiv **\*=** 10**;**

quotient **/=** 10**;**

**}**

//frontdiv /= 10;

**for** **(;** frontdiv **>** reardiv**;** frontdiv **/=** 10**,** reardiv **\*=** 10**)**

**{**

**if** **((**x **/** frontdiv**)** **%** 10 **!=** **(**x **/** reardiv**)** **%** 10**)**

**return** **false;**

**}**

**return** **true;**

**}**

**};**