Leetcode

题解

姓名：刘斯宇

学号：17341110

Contents

1 Two Sum

题目

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.

例子

Given nums = [2, 7, 11, 15], target = 9,
Because nums[0] + nums[1] = 2 + 7 = 9, return [0, 1].

思路

配对的题目，我们当然可以用两层额 for 循环来解决这个问题，但是这样的话，就没什么意义了，所以我们要降低时间复杂度，也就是要降低内层查找的时间复杂度，如何降低？用空间来换时间，所以很显然需要有个 map，这样就很容易的得到思路了。代码中比较巧妙的一点是如果出现了 vector 有相同的值的话，但是我们用后面的 induce 去覆盖前面的，这样我们遍历的时候就不会忘掉这种相同的值，但是 induce 不同的情况。

```
vector<int> twoSum(vector<int>& nums, int target)
{
    unordered_map<int,int>value2index;
    for(int i=0;i<nums.size();++i)
        value2index[nums[i]]=i;
    for(int i=0;i<nums.size();++i)
    {
        int temp=target-nums[i];
        if(value2index.find(temp)!=value2index.end()&&i!=value2index[temp])
            return {i,value2index[temp]};
    }
    return {};
}
```

2 Add Two Numbers

题目

You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order and each of their nodes contain a single digit. Add the two numbers and return it as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

例子

Input: (2 -> 4 -> 3) + (5 -> 6 -> 4)

Output: 7 -> 0 -> 8

Explanation: 342 + 465 = 807.

思路

链表的题目比较简单，一直向前遍历，直到找到结束的标志--两个链表都遍历完了以及没有进位了。

```
    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2)
    {
        ListNode*temp1=l1;
        ListNode*temp2=l2;
        ListNode*start=new ListNode(0);
        ListNode*mm=start;
        int jinwei=0;
        while(temp1||temp2||jinwei)
        {
            int sum=0;
            if(temp1)
            {
                sum+=temp1->val;
                temp1=temp1->next;
            }
            if(temp2)
            {
                sum+=temp2->val;
                temp2=temp2->next;
            }
            if(jinwei)
                sum+=1;
            if(sum>=10)
                jinwei=1;
            else
                jinwei=0;
            sum=sum%10;
            ListNode*temp=new ListNode(sum);
            start->next=temp;
            start=temp;
        }
        return mm->next;
    }
```

3 Longest Substring Without Repeating Characters

题目

Given a string, find the length of the longest substring without repeating characters.

例子

Input: "abcabcbb"

Output: 3

Explanation: The answer is "abc", with the length of 3.

Input: "bbbbb"

Output: 1

Explanation: The answer is "b", with the length of 1.

Input: "pwwkew"

Output: 3

Explanation: The answer is "wke", with the length of 3. Note that the answer must be a substring, "pwke" is a subsequence and not a substring.

思路

因为要找子序列，并且子序列不能有重复元素，所以一定是滑动窗口法，一个元素记录最左边+unordered_map 就可以了

```
    int lengthOfLongestSubstring(string s)
{
    if(s.empty())
        return 0;
    int left=0;
    int res=0;
    unordered_map<char,int>temp;
    for(int i=0;i<s.size();++i)
    {
        if (!temp[s[i]]||left>temp[s[i]])
        {
            res=max(res,i-left+1);
        }
        else
        {
            left=temp[s[i]];
        }
        temp[s[i]]=i+1;
    }
    return res;
}
```

4 Median of Two Sorted Arrays

题目

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)).

You may assume nums1 and nums2 cannot be both empty.

例子

nums1 = [1, 3]

nums2 = [2]

The median is 2.0

nums1 = [1, 2]

nums2 = [3, 4]

The median is (2 + 3)/2 = 2.5

思路

要求时间复杂度为对数型的，那么就一定要用算法，而且这样的搜索能用的似乎也就是二分法了，怎么二分呢，对什么二分呢？这就是考验功底的时候了，我们可以对我们要找的数字的 index 进行二分，分别在两个数列中取中间 index 的数值，两者间的较小的值的一部分被淘汰。以此不断的淘汰。

```cpp
int help(vector<int>& nums1, vector<int>& nums2,int number1,int number2,int target)
{
    if(number1>=nums1.size())
        return nums2[target+number2-1];
    if(number2>=nums2.size())
        return nums1[target+number1-1];
    if(target==1)
        return min(nums1[number1],nums2[number2]);
    int mid1=(number1+target/2-1>=nums1.size())?INT_MAX:nums1[number1+target/2-1];
    int mid2=(number2+target/2-1>=nums2.size())?INT_MAX:nums2[number2+target/2-1];
    if(mid1<mid2)
        return help(nums1,nums2,number1+target/2, number2, target-target/2);
    return help(nums1,nums2,number1, number2+target/2, target-target/2);
}
double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2)
{
    int num1=(int)nums1.size();
    int num2=(int)nums2.size();
    return (help(nums1,nums2,0,0,(num1+num2+1)/2)+help(nums1,nums2,0,0,(num1+num2+2)/2))/2.0;
}
```

5 Longest Palindromic Substring

题目

Given a string s, find the longest palindromic substring in s. You may assume that the maximum length of s is 1000.

例子

Example 1:

Input: "babad"

Output: "bab"

Note: "aba" is also a valid answer.

Example 2:

Input: "cbbd"

Output: "bb"

思路

对称子串，那么最容易想到的就是从某点出发从中间扩展了，但是要分子串的字符数量是奇数还是偶数的情况。

```
void help(int start1,int start2,int &start,int &maxlen,string&s)
{
    int len=(int)s.size();
    while(start1>=0&&start2<len&&s[start1]==s[start2])
    {
        --start1;++start2;
    }
    ++start1;--start2;
    if(maxlen<start2-start1+1)
    {
        maxlen=start2-start1+1;
        start=start1;
    }
}
string longestPalindrome(string s)
{
    int len=(int)s.size();
    int start=0,maxlen=0;
    for(int i=0;i<len;++i)
    {
        help(i, i, start, maxlen, s);
        help(i, i+1, start, maxlen, s);
    }
    return s.substr(start,maxlen);
}
```

6 Reverse Integer

题目

Given a 32-bit signed integer, reverse digits of an integer.

例子

Example 1:

Input: 123

Output: 321

Example 2:

Input: -123

Output: -321

Example 3:

Input: 120

Output: 21

思路

第一想法

```
```