代码随想录——算法训练营DAY1

数组

1. 特殊very special：连续内存地址

ACM模式输入输出参考

acm模式输入输出

LC 704 二分查找

题目链接：LC 704 二分查找
文章讲解：https://programmercarl.com/0704.%E4%BA%8C%E5%88%86%E6%9F%A5%E6%89%BE.html
视频讲解：https://www.bilibili.com/video/BV1fA4y1o715

类别

double ptr；双指针的左右指针，二分法

易错点

1. left, right初始值
2. while 里 left 与 right 可以取==?
3. 如何更新left right

其他：防止溢出, 没有直接(right+left)/2，　mid = left + (right - left) / 2

左闭右开[)

具体解释看代码，这里只是题纲，用来回忆

1. right = nums.size();
2. while (left < right)
3. right = mid，left = mid + 1

左闭右闭[]

1. right = nums.size()-1;
2. while (left < = right)
3. right = mid-1，left = mid + 1

ACM模式读入vector

```
 vector<int> nums;
    int num;
    while(cin >> num){
        nums.push_back(num);
        if(getchar() == '\n'){
            break;
        }
    }
```

ACM实现并输出类

```
cout<< Solution().search(nums,target) << ' ' << Solution2().search(nums,target)
<< endl;
```

code

```
#include<iostream>
#include<vector>
using namespace std;
int main(){
    class Solution{
    public:
        int search(vector<int>& nums, int target){
            // [) 左闭右开，所以右取不到
            int left = 0, right = nums.size();
            // 当left== right，区间[left, right)没效了，所以用 <
            while (left < right){
                int mid = left + (right - left) / 2; // 防止溢出，没有直接
(right+left)/2
                if (nums[mid] > target){
                    // target 在左半， 动right; 因为[) right取不到，mid已经比过了，所以
right= mid
                    right = mid;
                }
                else if (nums[mid] < target){
                    // target在右半
                    left = mid + 1;
                }
                else
                    return mid;
            }
            return -1;
        }
    };

    class Solution2{
    public:
        int search(vector<int>& nums, int target){
            // [] 左闭右闭，所以右可以取到，所以是-1，如果不减1，根本下标取不到
            int left = 0, right = nums.size() - 1;
            // 这里自己做的时候错了！！！！
            // 当left==right，区间[left, right]依然有效，所以用 <= 可以取==
            while (left <= right){
                int mid = left + (right - left) / 2; // 防止溢出，没有直接
(right+left)/2
```

```
                if (nums[mid] > target){
                    // target 在左半， 动right; 因为[] right取得到，mid已经比过了，所以
right= mid-1
                    right = mid -1;
                }
                else if (nums[mid] < target){
                    // target在右半
                    left = mid + 1;
                }
                else
                    return mid;
            }
            return -1;
        }
    };

    vector<int> nums;
    int num;
    while(cin >> num){
        nums.push_back(num);
        if(getchar() == '\n'){
            break;
        }
    }

    int target;
    cin>>target;

    cout<< Solution().search(nums,target) << ' ' <<
Solution2().search(nums,target) << endl;
    return 0;
}
```

LC 27 删除元素

题目链接：https://leetcode.cn/problems/remove-element/

文章讲解：
https://programmercarl.com/0027.%E7%A7%BB%E9%99%A4%E5%85%83%E7%B4%A0.html

视频讲解：https://www.bilibili.com/video/BV12A4y1Z7LP

类别

double ptr；双指针的fast slow ptr快慢指针

原地删除用双指针！！！！！！数组一定会影响内存地址

易错点

1. fast slow双指针；fast遍历old数组， slow记录新的数组
2. for (int fast = 0; fast <= nums.size()-1; fast++) 不行 nums.size()=0情况 nums=[] **这个只有自己真的按自己思路写过才知道，看答案不会有这种错误！！！**
3. for内if内 是谁和谁比，条件写不对

ACM模式读入vector

```
vector<int> nums;
    int num,target;
    while(cin>>num){
        nums.push_back(num);
        if (getchar() == '\n') break;
    }
```

ACM实现并输出类

```
cout<< Solution().removeElement(nums,target)<<endl;
```

code

```
#include<iostream>
#include<vector>
using namespace std;
class Solution {
public:
    int removeElement(vector<int>& nums, int val) {
        int slow = 0;
        // for (int fast = 0; fast <= nums.size()-1; fast++){ //这样写， nums=[]
就会报错，runtime error: reference binding to null pointer of type 'int'
(stl_vector.h)； 因为nums.size()此时=0 nums.size()-1=-1就有问题了
        for (int fast = 0; fast < nums.size(); fast++){
            if (val != nums[fast]){
                // 如果不等于,那么不删不跳过,所以赋值； 第一次写错了，把nums[fast]和
nums[slow]比，脑子没想清楚谁和谁比！！
                nums[slow++] = nums[fast];
            }
        }
        return slow;
    }
};
int main(){
    vector<int> nums;
    int num,target;
    while(cin>>num){
        nums.push_back(num);
        if (getchar() == '\n') break;
    }
    cin>>target;
    cout<< Solution().removeElement(nums,target)<<endl;
    return 0;
}
```

拓展 DAY2 复习了

LC 35 没做完！看情况！！

题目：https://leetcode.cn/problems/search-insert-position/

文章：https://programmercarl.com/0035.%E6%90%9C%E7%B4%A2%E6%8F%92%E5%85%A5%E4%BD%8D%E7%BD%AE.html#%E6%80%9D%E8%B7%AF

类别

double ptr之左右指针; 704的拓展

易错点

1. **只要看到面试题里给出的数组是有序数组，都可以想一想是否可以使用二分法**
2. 分类讨论情况不全!

   没打勾的根本没想到！！，return mid+1想错了，why right+1， 草稿纸上做一下就知道了！！！
   写出left right mid

   - [ ] 目标值在数组所有元素之前
   - [x] 目标值等于数组中某一个元素
   - [x] 目标值插入数组中的位置 （想到了，but return什么想不到）
   - [ ] 目标值在数组所有元素之后

code:

错误

```
class Solution {
public:
    int searchInsert(vector<int>& nums, int target) {
        // []
        int left = 0, right = nums.size()-1;
        int result;
        while(left <= right){
            int mid = left + (right - left) / 2;
            if(nums[mid] > target){
                //target in the left
                right = mid - 1;
            }
            else if(nums[mid] < target){
                // target in the right
                left = mid + 1;
            }
            else return mid;
            result = mid;
        }
        return result + 1; //这里有问题！！
    }
};
```