剑指Offer（六）：旋转数组的最小数字

2017年11月24日 10:22:41 8 7,287 °C 编辑

一、前言

本系列文章为《剑指Offer》刷题笔记。

刷题平台：牛客网

书籍下载：共享资源

二、题目

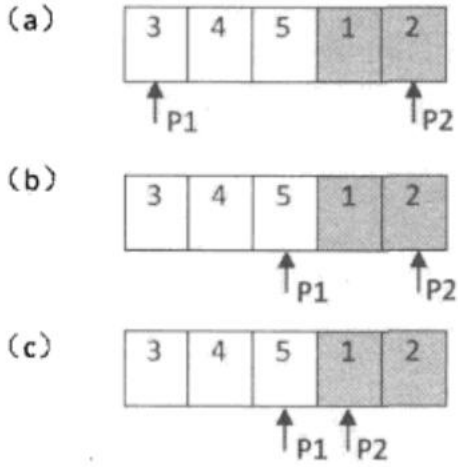
把一个数组最开始的若干个元素搬到数组的末尾，我们称之为数组的旋转。 输入一个非递减排序的数组的一个旋转，输出旋转数组的最小元素。 例如组{3,4,5,1,2}为{1,2,3,4,5}的一个旋转，该数组的最小值为1。 NOTE：给出的所有元素都大于0，若数组大小为0，请返回0。

1、思路

我们注意到旋转之后的数组实际上可以划分为两个排序的字数组，而且前面的字数组的元素大于或者等于后面字数组的元素。我们还注意到最小的元素刚好是这两个字数组的分界线。在排序的数组中可以用二分查找实现O(logn)的查找。本题给出的数组在一定程度上是排序的，因此我们可以试着用二分查找法的思路来寻找这个最小的元素。

- 把一个数组最开始的若干个元素搬到数组的末尾，我们称之为数组的旋转。 输入一个非递减排序的数组的一个旋转，输出旋转数组的最小元素。例如数组{3,4,5,1,2}为{1,2,3,4,5}的一个旋转，该数组的最小值为1。 NOTE：给出的所有元素都大于0，若数组大小为0，请返回0。
- 接着我们可以找到数组中间的元素。如果中间元素位于前面的递增子数组，那么它应该大于或者等于第一个指针指向的元素。此时最小元素应该位于该中间元素之后，然后我们把第一个指针指向该中间元素，移动之后第一个指针仍然位于前面的递增子数组中。
- 同样，如果中间元素位于后面的递增子数组，那么它应该小于或者等于第二个指针指向的元素。此时最小元素应该位于该中间元素之前，然后我们把第二个指针指向该中间元素，移动之后第二个指针仍然位于后面的递增子数组中。
- 第一个指针总是指向前面递增数组的元素，第二个指针总是指向后面递增数组的元素。最终它们会指向两个相邻的元素，而第二个指针指向的刚好是最小的元素，这就是循环结束的条件。

示意图如下：



特殊情况：

- 如果把排序数组的0个元素搬到最后面，这仍然是旋转数组，我们的代码需要支持这种情况。如果发现数组中的一个数字小于最后一个数字，就可以直接返回第一个数字了。
- 下面这种情况，即第一个指针指向的数字、第二个指针指向的数字和中间的数字三者相等，我们无法判断中间的数字1是数以前面的递增子数组还是后面的递增子数组。正样的话，我们只能进行顺序查找。

2、代码

C++:

```C+
class Solution {
public:
    int minNumberInRotateArray(vector<int> rotateArray) {
        int size = rotateArray.size();                              //数组长度
        if(size == 0){
            return 0;
        }
        int left = 0;                                              //左指针
        int right = size - 1;                                      //右指针
        int mid = 0;                                               //中间指针
        while(rotateArray[left] >= rotateArray[right]){             //确保旋转
            if(right - left == 1){                                 //左右指针相邻
                mid = right;
                break;
            }
            mid = left + (right - left) / 2;                      //计算中间指针位置
            //特殊情况：如果无法确定中间元素是属于前面还是后面的递增子数组，只能顺序查找
            if(rotateArray[left] == rotateArray[right] && rotateArray[mid] == rotateArray[left]){
                return MinInOrder(rotateArray, left, right);
            }
            //中间元素位于前面的递增子数组，此时最小元素位于中间元素的后面
            if(rotateArray[mid] >= rotateArray[left]){
                left = mid;
            }
            //中间元素位于后面的递增子数组，此时最小元素位于中间元素的前面
            else{
                right = mid;
            }
        }
        return rotateArray[mid];
    }
private:
    //顺序寻找最小值
    int MinInOrder(vector<int> &num, int left, int right){
        int result = num[left];
        for(int i = left + 1; i < right; i++){
            if(num[i] < result){
                result = num[i];
            }
        }
        return result;
    }
};
```

Python2.7：

```Pytho
# -*- coding:utf-8 -*-
```

```python
class Solution:
    def minNumberInRotateArray(self, rotateArray):
        # write code here
        if len(rotateArray) == 0:
            return 0
        left = 0
        right = len(rotateArray) - 1
        mid = 0
        while rotateArray[left] >= rotateArray[right]:
            if right - left == 1:
                mid = right
                break
            mid = left + (right - left) // 2
            if rotateArray[left] == rotateArray[mid] and rotateArray[mid] == rotateArray[right]:
                return self.minInorder(rotateArray, left, right)
            if rotateArray[mid] >= rotateArray[left]:
                left = mid
            else:
                right = mid
        return rotateArray[mid]

    def minInorder(self, array, left, right):
        result = array[left]
        for i in range(left+1, right+1):
            if array[i] < result:
                result = array[i]
        return result
```


简洁是智慧的灵魂，冗长是肤浅的藻饰。--- 莎士比亚