Find Minimum in Rotated Sorted Array II

> *Follow up* for "Find Minimum in Rotated Sorted Array":
> What if *duplicates* are allowed?
>
> Would this affect the run-time complexity? How and why?

Suppose a sorted array is rotated at some pivot unknown to you beforehand.

(i.e., `0 1 2 4 5 6 7` might become `4 5 6 7 0 1 2`).

Find the minimum element.

The array may contain duplicates.

Solution 1

```
class Solution {
public:
    int findMin(vector<int> &num) {
        int lo = 0;
        int hi = num.size() - 1;
        int mid = 0;

        while(lo < hi) {
            mid = lo + (hi - lo) / 2;

            if (num[mid] > num[hi]) {
                lo = mid + 1;
            }
            else if (num[mid] < num[hi]) {
                hi = mid;
            }
            else { // when num[mid] and num[hi] are same
                hi--;
            }
        }
        return num[lo];
    }
};
```

When num[mid] == num[hi], we couldn't sure the position of minimum in mid's left or right, so just let upper bound reduce one.

written by sheehan original link here

Solution 2

```
class Solution {
public:
    int findMin(vector<int> &num) {
        if(num.empty())
            return 0;
        int i=0,j=num.size()-1;
        while(i<j)
        {
            int mid=(i+j)/2;
            if(num[j]<num[mid]){
                i=mid+1;
            }
            else if(num[mid]<num[j]){
                j=mid;
            }
            else{//num[mid]==num[j]
                if(num[i]==num[mid]){//linear complexity
                    i++;
                    j--;
                }
                else
                    j=mid;
            }
        }
        return num[j];
    }
};
```

written by kmind original link here

Solution 3

```
public int findMin(int[] nums) {
     int l = 0, r = nums.length-1;
     while (l < r) {
         int mid = (l + r) / 2;
         if (nums[mid] < nums[r]) {
             r = mid;
         } else if (nums[mid] > nums[r]){
             l = mid + 1;
         } else {
             r--;  //nums[mid]=nums[r] no idea, but we can eliminate nums[r];
         }
     }
     return nums[l];
}
```

written by jinwu original link here