< Back

Python: Two Ways, O(N) and O(logN)

14

dibdidib ★ 158 Last Edit: December 30, 2019 4:53 AM 1.1K VIEWS

During a contest, we see that we have only a small number of elements, so we can take the quick route with time complexity `O(N)`:

```python
def isMajorityElement(self, nums, target):
    return nums.count(target) > len(nums) // 2
```

During an interview, we would think a bit more. Letting `x` be our majority element, a sorted array containing a majority element would look like one of the following:

```
[ x x x x x x x . . . . . . ]        # majority at the beginning
[ . . . x x x x x x x . . . ]        # majority at the middle
[ . . . . . . x x x x x x x ]        # majority at the ending
```

If there is a majority element then when we examine the middle index of the array, we are guaranteed to find the majority element. So we can binary search for the beginning and end of the streak:

```python
def isMajorityElement(self, nums, target):
    N = len(nums)
    if nums[N // 2] != target:
        return False
    lo = bisect.bisect_left(nums, target)
    hi = bisect.bisect_right(nums, target)
    return hi - lo > N // 2
```

This cuts our time complexity from `O(N)` to `O(logN)`. If an interviewer asks us to do it without `bisect`, we can spell out the logic of `bisect_left()` and use a helper to find the required indices. This will likely run a bit slower because it is pure Python, and the `bisect` module typically uses a faster underlying implementation in C:

```python
def isMajorityElement(self, nums, target):

        def search(a, x):
            lo, hi = 0, len(a)
            while lo < hi:
                mid = (lo + hi) // 2
                if a[mid] < x:
                    lo = mid + 1
                else:
                    hi = mid
            return lo

        N = len(nums)
        if nums[N // 2] != target:
            return False
        lo = search(nums, target)
        hi = search(nums, target + 1)
        return hi - lo > N // 2
```

Thanks to @ye15 for pointing out that we can apply a similar approach to use only one binary search:

```python
def isMajorityElement(self, nums, target):
    k = len(nums) // 2
    if nums[k] != target:
        return False
    lo = bisect.bisect_left(nums, target, 0, k)
    hi = lo + k
    return hi < len(nums) and nums[hi] == target
```

python binary search python 3

Comments: 4 Best Most Votes Newest to Oldest Oldest to Newest

Type comment here... (Markdown is supported)