153. Find Minimum in Rotated Sorted Array (Medium)

You are given an array of length `n` which was originally sorted in ascending order. It has now been rotated between `1` and `n` times. For example, the array `nums = [1,2,3,4,5,6]` might become:

`[3,4,5,6,1,2]` if it was rotated `4` times. `[1,2,3,4,5,6]` if it was rotated `6` times.

- `[3,4,5,6,1,2]` if it was rotated `4` times.
- `[1,2,3,4,5,6]` if it was rotated `6` times.

Notice that rotating the array `4` times moves the last four elements of the array to the beginning. Rotating the array `6` times produces the original array.

Assuming all elements in the rotated sorted array `nums` are unique, return the minimum element of this array.

A solution that runs in `O(n)` time is trivial, can you write an algorithm that runs in `O(log n) time`?

Example 1:

```
Input: nums = [3,4,5,6,1,2]

Output: 1
```

Example 2:

```
Input: nums = [4,5,0,1,2,3]

Output: 0
```

Example 3:

```
Input: nums = [4,5,6,7]
```

```
Output: 4
```

Constraints:

- `1 <= nums.length <= 1000`
- `1000 <= nums[i] <= 1000`

▼ 思路:

作法一：紀錄比對r與mid，確認最小值範圍

T:$O(n)$, S:$O(1)$

作法一

```
class Solution {
public:
    int findMin(vector<int> &nums) {
        int l=0, r=nums.size()-1,mid,ans=INT_MAX;
        while(l<r){
            mid = (l+r)/2;
            if(nums[mid] > nums[r]){
                l = mid+1;
            }
            else{
                r = mid;
            }
        }
        return nums[l];
    }
};
```