Strivers-A2Z-DSA-Sheet-main\02.Binary Search\1D Arrays\11.Find_the_minimum_element_in_sorted_rotated_array.cpp

```
/*
QUESTION:
Suppose an array of length n sorted in ascending order is rotated between 1 and n times. For example, the array nums = [0,1,2,4,5,6,7] might become:

[4,5,6,7,0,1,2] if it was rotated 4 times.
[0,1,2,4,5,6,7] if it was rotated 7 times.

Given the sorted rotated array nums of unique elements, return the minimum element of this array.

APPROACH:
We can use the binary search approach to find the minimum element.
1. Initialize low = 0 and high = n-1, where n is the size of the array.
2. While low < high, calculate mid = low + (high - low) / 2.
3. If nums[mid] > nums[high], it means the minimum element is on the right side of mid, so update low = mid+1.
4. Otherwise, the minimum element is on the left side of mid or mid itself, so update high = mid.
5. After the loop ends, low will be pointing to the minimum element index.
6. Return nums[low] as the result.

CODE:
*/

int findMin(vector<int>& nums) {
    int low = 0, high = nums.size()-1;
    while(low < high){
        int mid = low + (high - low) / 2;
        if(nums[mid] > nums[high])
            low = mid+1;
        else
            high = mid;
    }
    return nums[low];
}

// TIME COMPLEXITY: O(log n)
// SPACE COMPLEXITY: O(1)

```