Daily Coding Problem #203

Problem

This problem was asked by Uber.

Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand. Find the minimum element in O(log N) time. You may assume the array does not contain duplicates.

For example, given [5, 7, 10, 3, 4], return 3.

Solution

The O(log N) in the instructions may tip you off to the fact that we can use binary search to solve this problem.

Let's look at what happens if we split our list at some point. If the value at our split point is less than the last element in the list, we know the right half is sorted, so the minimum element must lie in the left half (including the midpoint). Otherwise, it must lie in the right half.

We can apply this routine repeatedly to cut in half the area where the minimum element must be. Eventually we will reach a point where the lowest and highest indices we are looking at will be the same, and the element at this index will be our solution.

```
def helper(arr, low, high):
    if high == low:
        return arr[low]

    mid = (high + low) / 2
    if arr[mid] < arr[high]:
        high = mid
    else:
        low = mid + 1
```

```
    return helper(arr, low, high)

def find_min_element(arr):
    low, high = 0, len(arr) - 1
    return helper(arr, low, high)
```