Problem 2: Search in a Rotated Sorted Array

To solve the problem, First I look up for the pivot point, divide the array into two sub arrays and do binary search on both sub arrays. To find the pivot i call a recursive function which in the worst case adds O(n) time complexity. Second, we do binary search in the sub arrays so in the worst case the binary search adds O(log n) time complexity.
We finally end up with O(n) + O(log n) which can be just simplified as O(log n).

Time and Space complexity
Time → O (log n)
Space → O(1), because we return a single value