**33. Search in Rotated Sorted Array**

Medium

Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand.

(i.e., [0,1,2,4,5,6,7] might become [4,5,6,7,0,1,2]).

You are given a target value to search. If found in the array return its index, otherwise return -1.

You may assume no duplicate exists in the array.

Your algorithm's runtime complexity must be in the order of *O*(log *n*).

**Example 1:**

**Input:** nums = [4,5,6,7,0,1,2], target = 0

**Output:** 4

**Example 2:**

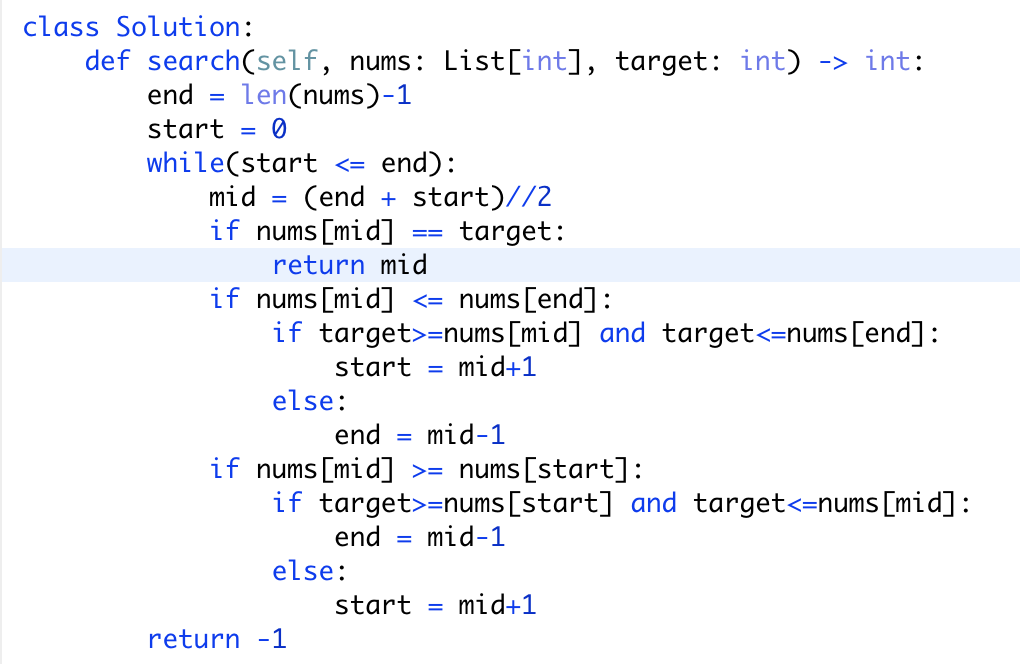
**Input:** nums = [4,5,6,7,0,1,2], target = 3

**Output:** -1

这道题的难点在于：首先不知道是在哪个点rotated的。其次时间复杂度要求是logn，这就必须要用到二分搜索。

思路：始终寻找有序的那段数列

该算法是在二分搜索上改良的一个算法。我们将中间的数分别和头尾的数进行比较，去寻找始终是升序的那段数列。比如[4,5,6,7,0,1,2], mid=7，start=4，end=2，有序的一段数列是从4到7（始终升序），而7到2（包括7）是无序的（7到0降序，0到2升序）。**此时可以得出结论：当start<mid的时候，左半段是有序的**。再比如[5,6,7,0,1,2,4], mid=0，start=5，end=4，有序的一段数列是从0到4（始终升序），而5到0（包括7）是无序的（5到7升序，7到0降序）。**此时可以得出结论：当mid<end的时候，右半段是有序的**。选取有序的数列后就可以依据两端的数字大小判断target是否在这段数字里面，在就选择有序数列继续做二分搜索，不在就选无序数列重复上述步骤，依据上面的结论选取左或右半段进行同样的分析。



Tips：>=, <=还是>, <的使用（是否遍历到所有元素）