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.

题目：

就是给一个排好序的顺序表（数组），然后进行查找。但是这个顺序表会向滚带一样动

0,1,2,4,5,6,7 🡪4,5,6,7,0,1,2

思路：

首先看到有序的顺序表查询，最先想到的应该是二分查询。采用二分查询，需要考虑的便是mid怎么取值。

要是0,1,2,4,5,6,7的原始数列

low=0，high=nums.length

最开始分区的mid取中间即可mid=(low+high)/2

然后进行分区的查询

要是4,5,6,7,0,1,2的变动的数列呢？

我们看到当递增结束的那个数字，也就是当前数列的7，7之前是递增，7之后同样是递增，只不过7和0天然的就被分成两块。一块4到7，一块0到2.

同理别的变动数组也一样，同样会被天然分成两个部分，那么此时的mid就应该等于7这个位置。

所以我们处理变动数列的时候，要找到7这个位置，然后领mid等于这个位置，就能成功的把数组分成两块。

当时还要注意，当分块完成的时候，用key跟mid比价确定分块的时候，你需要加个条件，因为7是最大的数字，你怎么比较都是小于7的，你要留意左边的最小的数字，这个数列也就是4，只有key大于4，小于7，才会被分到第一块，否则分到另一个块中。然后进行二分法的比较。

代码：

public static int search(int[] nums,int key){

int low=0,high=nums.length; //初始化

int mid = (low+high)/2;//mid初始化要是数组为递增顺序mid=(low+high)/2

int i=0;

while(i<nums.length-2){ //遍历数组，查看数组是否为递增顺序

if(nums[i]>nums[i+1]) //若不是，修改mid值为递增顺序最后一个数字

mid=i;

i++;

}

while(low<=high){ //二分查找

if(key==nums[mid])//因为之前定义了mid的值，所以这么先进行对比再修改mid值

return mid; //找到返回下标mid

if(key<nums[mid]&&key>=nums[low]) //确定key所在的区间，注意这里加了比较low的大小，纯自增数列不用

high=mid-1;

else

low=mid+1;

mid=(low+high)/2;

}

return -1; //没有查询到，查询失败，返回-1

}

体会：这道题看题，有序数组查询就应该直接相当用二分法，然后再根据题目将二分法适配题目。