Given an array of integers sorted in ascending order, find the starting and ending position of a given target value.

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

If the target is not found in the array, return [-1, -1].

For example,  
Given [5, 7, 7, 8, 8, 10] and target value 8,  
return [3, 4].

题目：

还是对有序的数列的静态查找，这次查找的是范围。

例如，找8，8开始的位置3，结束的位置是4.

对有序的数组进行查找，我们首先要想到二分法。

这道题跟普通二分法是有差别的，因为单次的二分法查找，遇到相等的就返回当前位置即可。这题不一样，遇到当前相等的还需要继续判断，例如，找下届，遇到相等的还需要继续往下进行，直到找到第一个等于target的数字为止。

public static int[] searchRange(int[] nums, int target) {

int[] targerRange={-1,-1}; //初始化range

int low =0 ,high = nums.length-1;

while(low<high){ //确定target的下届

int mid = (low+high)/2;

if(target<=nums[mid]){ //当小于的时候high=mid，当等于的时候也要向前走，因为要确定下届。

high=mid-1;

}

else //当大于mid的时候，low向前走，注意low一定不能等于mid，因为mid是向下取整的，容易造成死循环

low=mid+1;//每次low走都要等于mid+1

if(low==high) //当low=high的时候，确立一个位置。

targerRange[0]=high;

}

low=0;

high=nums.length-1;

while(low<high){ //确定上届

int mid = (low+high)/2;

if(target>=nums[mid]) //当大于等于的时候，low向前走。

low=mid+1;

else //当小于的时候折半找high的位置

high=mid-1;

if(high==low) //因为high一直是在不等于targer的时候走的，所以high走到的位置是target的下一位

targerRange[1]=high-1;

}

return targerRange;

}

当时有个问题就是头部和尾部没有办法判断，需要添加一个尾部判断的方法。

If(nums[0]==target) targerRange[0] =0;

If(nums[nums.length]==target) targerRange[1]=nums.length;

方法二：

从正面遍历数组，遇到第一个等于target的数字停下，返回位置。

从后边再次遍历数组，遇到第一个等于target的数字停下，返回位置。