

思路：SORTED+LOGN, binary Search

具体思路：因为我们要找到first和last,那么两种searching模式是不一样的，first每次循环如果左边还剩，取左半截，last右边还剩，取右半截，

-1 first,0 search,+1 last

一开始mid先找有没有这个数，

没有，return -1-1

有左边找first,右边找last

first如果mid等于

左边还有相同的target，找左边,

左边没有相同的，return mid

不然搜右边

BinarySearch关键点：

第一步mid=left+right/2

Base case

两种

left>right, return false

nums[mid]=target, return mid

如果nums[mid]<target,说明target在右边，循环mid+1,right

如果nums[mid]>target,说明target在左边，循环left，mid-1,

+1-1很重要，不然会无限循环

例如只剩两个的时候

target在第二个，然后我们循环mid,right，那么target还是在第二个，只有mid+1,right，这时剩一个，成功

class Solution {

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

int[] result=new int[2];

binarySearch(nums,target,result,0,nums.length-1,0);

return result;

}

public void binarySearch(int[] nums,int target, int[] result,int left, int right,int priority){

int mid=(left+right)/2;

if(priority==0)

{

if(left>right){

result[0]=-1;

result[1]=-1;

return;

}

if(nums[mid]==target)

{

binarySearch(nums,target,result,left,mid,-1);

if(mid==nums.length-1||nums[mid+1]>target)

result[1]=mid;

else

binarySearch(nums,target,result,mid+1,right,1);

}

else if(nums[mid]>target)

binarySearch(nums,target,result,left,mid-1,0);

else if(nums[mid]<target)

binarySearch(nums,target,result,mid+1,right,0);

}

if(priority==-1){

if(nums[mid]==target){

if(mid!=0&&nums[mid-1]==target)

binarySearch(nums,target,result,left,mid,-1);

else if(mid==0||nums[mid-1]!=target){

result[0]=mid;

return;

}

}

else

binarySearch(nums,target,result,mid+1,right,-1);

}

if(priority==1){

if(nums[mid]==target){

if(mid!=nums.length-1&&nums[mid+1]==target)

binarySearch(nums,target,result,mid+1,right,1);

else if(mid==nums.length-1||nums[mid+1]!=target){

result[1]=mid;

return;

}

}

else

binarySearch(nums,target,result,left,mid,1);

}

}

}