/\*

Follow up for "Search in Rotated Sorted Array":

What if duplicates are allowed?

Would this affect the run-time complexity? How and why?

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).

Write a function to determine if a given target is in the array.

way-1:先去重排序，再二分查找

way-2: 这个方法不排序去重，

由于有重复元素，a[m]>=a[l],[l,m]是递增元素的条件就不成立。

如果a[m]=a[l],l++

\*/

class Solution {

public:

bool search(vector<int>& nums, int target)

{

//way-1

/\*

vector<int>::iterator it;

it=unique(nums.begin(),nums.end());

nums.resize(it-nums.begin());

int l=0;

int r=nums.size()-1;

while(l<=r)

{

int m=(r+l)/2;

if(nums[m]==target)

return true;

else if(nums[l]<=nums[m])

{

if(target>=nums[l] && target<nums[m])

r=m-1;

else

l=m+1;

}

else

{

if(target>=nums[l] || target<nums[m])

r=m-1;

else

l=m+1;

}

}

return false;

\*/

//way-2

int l=0;

int r=nums.size()-1;

while(l<=r)

{

int m=(r+l)/2;

if(nums[m]==target)

return true;

else if(nums[l]<nums[m])

{

if(target>=nums[l] && target<nums[m])

r=m-1;

else

l=m+1;

}

else if(nums[l]==nums[m])//关键

{

l++;

}

else

{

if(target>=nums[l] || target<nums[m])

r=m-1;

else

l=m+1;

}

}

return false;

}

};