Given an unsorted integer array, find the first missing positive integer.

For example,  
Given [1,2,0] return 3,  
and [3,4,-1,1] return 2.

Your algorithm should run in *O*(*n*) time and uses constant space.

题目：找到数列缺失的正整数，当时数列是无序的。

缺失的地方的数字是0或者负数

我们要想到的是，如要要考虑缺失的当个数字，那么这个数列无论有序无序，它都是连续的，当连续的时候才可能出现缺失的单个数字。

然后我们需要做的是把数列排序，这里采用无序序列最好用的快速排序排好，因为确实的是单个数字，这个单个数字一定在排好序的序列最头。

我们略过最头的数据，开始遍历后边，如果每一位数据+1等于后一位，满足连续条件，如果不等于，那么不连续说明数字缺失。我们输出当前不连续位的数值+1，就是缺失的数字。如果整个排好序的数列都连续，缺失的是哪个呢？首先要是整个序列都连续，缺失的只有队首或者队尾，如果题目要求每个都是以1开头的话，我们直接就可以决定是队尾数据缺失了，因为比1小的数字正整数没有。如果不是以1开头呢，我们可以设置一个标志位来记录负数所在位置在没排序前是队首还是队尾，然后判定。

如果说，题目的恒定空间不允许在设置标志位，我们只能一个一个换元素。但是还元素有个问题就是你起始点不一定是1啊，要是1我的快速排序也可以啊。

我们这里复习一下，快速排序的时间复杂度为O(n),本次的空间复杂度为O(1),不稳定。

下面是代码

public static int partition(int[] nums,int i,int j){

int piovt = nums[i];

while(i<j){

if(i<j&&nums[j]>piovt)

j--;

else

nums[i]=nums[j];

i++;

if(i<j&&nums[i]<piovt)

i++;

else

nums[j]=nums[i];

j--;

}

nums[i]=piovt;

return i;

}

public static void quickSort(int[] nums,int low,int high){

if(low<high) {

int pivotloc = partition(nums, low, high);

quickSort(nums, low, pivotloc - 1);

quickSort(nums, pivotloc + 1, high);

}

}

public static int firstMissPositive(int[] nums){

quickSort(nums,0,nums.length-1);

for(int i=1;i<nums.length-1;i++){

if(nums[i]+1==nums[i+1])

continue;

else

return nums[i]+1;

}

if(nums[1]==1)

return nums[nums.length-1]+1;

else

return nums[1]-1;

}

但是题目要求是时间复杂度为O(n)，我们好像只能是换位置了

public static int firstMissPositive(int[] nums){

for(int i=0;i<nums.length;i++){

//把正整数移动到应该在的位置

//例如nums[i]=5，移动到4的位置，也就是nums[4],nums[i]-1=4;

if(nums[i]>0&&nums[i]<nums.length&&nums[i]!=nums[nums[i]-1]){

int temp = nums[i]; //两个位置互换

nums[i] = nums[nums[i]-1];

nums[nums[i]-1] = temp;

}

}

for(int i=0;i<nums.length;i++){ //遍历当遇到不连续的位置+1为最终结果

if(nums[i]!=i+1)

return i+1;

}

return nums.length+1;//因为1开头，当都连续时候，缺失的一定是队尾的那个数字

}

这里第一次遍历的时间复杂度为O(n),第二次遍历的时间复杂度为O(n),总的时间复杂度为O(2n)=O(n),满足题目要求。