Implement next permutation, which rearranges numbers into the lexicographically next greater permutation of numbers.

If such arrangement is not possible, it must rearrange it as the lowest possible order (ie, sorted in ascending order).

The replacement must be in-place, do not allocate extra memory.

Here are some examples. Inputs are in the left-hand column and its corresponding outputs are in the right-hand column.  
1,2,3 → 1,3,2  
3,2,1 → 1,2,3  
1,1,5 → 1,5,1

题目的意思是找到比当前序列字典顺序大1的那个序列。

首先我们来解释一下字典顺序吧。

在数学中，字典或词典顺序（也称为词汇顺序，字典顺序，字母顺序或词典顺序）是基于字母顺序排列的单词按字母顺序排列的方法。 这种泛化主要在于定义有序完全有序集合（通常称为字母表）的元素的序列（通常称为计算机科学中的单词）的总顺序。

对于数字1、2、3......n的排列，不同排列的先后关系是从左到右逐个比较对应的数字的先后来决定的。例如对于5个数字的排列 12354和12345，排列12345在前，排列12354在后。按照这样的规定，5个数字的所有的排列中最前面的是12345，最后面的是 54321。

就是123，132，213，231，312，321.

首先变的是后边的数字，最后变的是前边的数字。

例如1，5，8，4，7，6，5，3，1

算法：首先定义i为nums[nums.length-2],也就是3的位置。分别与前一位比较，如果大于等于前一位向下进行，找到第一个比前一位小的数字，就是4的位置。因为之前都是i比i+1要大，所以4之前都是逆序排列的。然后再从后边开始寻找第一个比4这个数字大的数字，也就是5。然后交换4和5，这样就完成了最小程度的替换。然后需要做的是将这串数字尽可能的小。怎么做？就是将那串逆序排列的数字倒过来变成正序。

代码：

public class NextPermutation {

public static void nextPermutation(int[] nums) {

int i = nums.length - 2;

while (i >= 0 && nums[i] >= nums[i + 1]) { //找到i的位置,i的位置正好是从后边起逆序的终点

i--;

}

if (i >= 0) {

int j = nums.length - 1;

while (j >= 0 && nums[j] <= nums[i]) { //找j的位置,j的位置是后边逆序子数列中比nums[i]大最小的数

j--; //也就是第一个比nums[i]大的数字

}

swap(i,j,nums); //交换i，j位置的两个数字

}

reverse(i+1,nums); //将交换后的数列变的尽可能的小，就需要把后边逆序的子序列变成正序

}

public static void swap(int i,int j,int[] nums){ //i，j两个位置数字交换

int temp = nums[i];

nums[i]=nums[j];

nums[j]=temp;

}

public static void reverse(int start,int[] nums){ //从start开始到结尾，逐个交换

int i = start; //nums[start]与nums[nums.length-1]换

int j = nums.length-1; //nums[start+1]与nums[nums.length-2]换，以此类推

while (i<j){ //到i=j的时候也就是中间的时候停止交换

swap(i,j,nums);

i++;

j--;

}

}

}