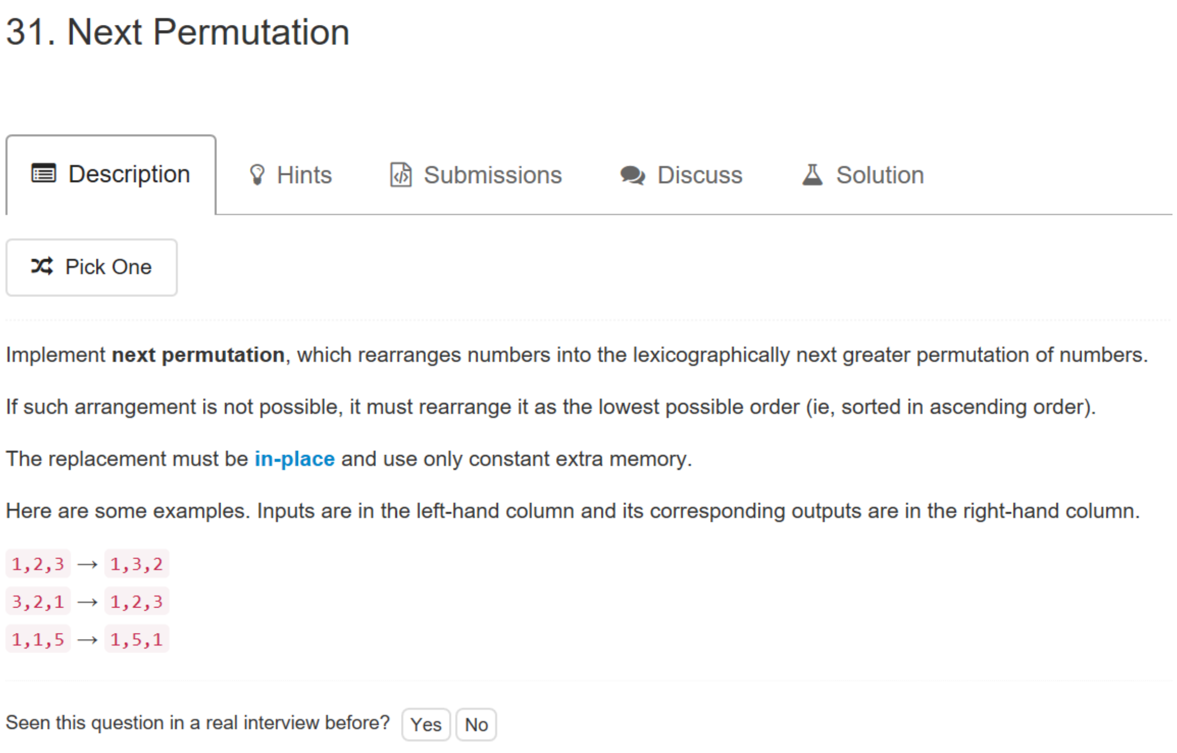

31. Next Permutation

Description | Hints | Submissions | Discuss | Solution

Pick One

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** and use only constant 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`

Seen this question in a real interview before? Yes No

```java
public class L31 {
   /* 这个题就能马上理清思路。下面我们用一个例子来说明，比如排列是(2,3,6,5,4,1)，求下一个排列的基本步骤是这样：
1) 先从后往前找到第一个不是依次增长的数，记录下位置p。比如例子中的3，对应的位置是1;
2) 接下来分两种情况：
   (1) 如果上面的数字都是依次增长的，那么说明这是最后一个排列，下一个就是第一个，其实把所有数字反转过来即可
   (比如(6,5,4,3,2,1)下一个是(1,2,3,4,5,6));
   (2) 否则，如果p存在，从p开始往后找，找找找，找到第一个比他小的数(个人认为是第一个不比它小的前面那个)，然后两个调换位置，比如例子中的4。
 调换位置后得到(2,4,6,5,3,1)。最后把p之后的所有数字倒序，比如例子中得到(2,4,1,3,5,6)， 这个即是要求的下一个排列。*/
     public void nextPermutation(int[] nums) {
         if(nums.length == 0 || nums == null) return ;
         int i = nums.length - 2;
         while (i >= 0 && nums[i] >= nums[i + 1])
            i --;
         if(i >= 0) {
             int j = i + 1;
             while (j < nums.length && nums[j] > nums[i])
                j ++;
             j --;
             swap(nums, i, j);
         }
         reverse(nums, i+1, nums.length - 1);
     }
    public void swap(int [] nums, int i, int j) {
        int tmp = nums[i];
        nums[i] = nums[j];
        nums[j] = tmp;
    }
    public void reverse(int [] nums, int i, int j) {
        while(i < j)
            swap(nums, i++, j--);
    }
}
```