非递减数列

给你一个长度为 n 的整数数组，请你判断在 最多 改变 1 个元素的情况下，该数组能否变成一个非递减数列。

我们是这样定义一个非递减数列的： 对于数组中所有的 i (0 <= i <= n-2)，总满足 nums[i] <= nums[i + 1]。

示例 1:

输入: nums = [4,2,3]

输出: true

解释: 你可以通过把第一个 4 变成 1 来使得它成为一个非递减数列。

示例 2:

输入: nums = [4,2,1]

输出: false

解释: 你不能在只改变一个元素的情况下将其变为非递减数列。

解题思路：
根据题意，破坏非递减数组序列的结构是‘N’字型结构。具体解法可以设置双指针，分别从数组两端往中间搜索，当碰到失序地方的时候停下来。如下图所示：

若至多改变一个元素能将原数组调整有序，当只有一处失序存在时必定有 high = low + 1,当数组完全符合非递减时 high = low,因此 high-low<=1 是命题的必要条件。
另一方面如图所示，当失序的时候有两种调整方案：
1.改变位置 low,使得 nums[low-1]<nums[low]<nums[low+1];
2.改变位置 high,使得 nums[high-1]< nums[high]<nums[high+1];
采用方案 1 前提条件为 nums[low+1]>=nums[low-1], 采用方案 2 前提条件为 nums[high+1]>=nums[high-1]。
再考虑两种端点情况，即在最开始和最结尾失序，low=0 或者 high=len-1 时，一定可以调整。

```
bool checkPossibility(int* nums, int numsSize){
```

```
    if(numsSize<=2)
        return true;
    int low=0;
    int high=numsSize-1;
    while(low<high)
    {
        while(low<high&&nums[low]<=nums[low+1])//符合非递减  low 后移
        {
            low++;
        }
        while(low<high&&nums[high]>=nums[high-1])//符合非递减  high 前移
        {
            high--;
        }
        return (high-low<=1&&(low==0||high==numsSize-
1||nums[low+1]>=nums[low-1]||nums[high-1]<=nums[high+1]));
    }
    return false;
}
```