/\*

Given an array with n objects colored red, white or blue, sort them so that objects of the same color are adjacent, with the colors in the order red, white and blue.

Here, we will use the integers 0, 1, and 2 to represent the color red, white, and blue respectively.

Note:

You are not suppose to use the library's sort function for this problem.

way-1:统计 012的个数，往里面放

way-2:

如果只能扫一遍，很容易想到的就是左边存放0和1，右边存放2.两边往中间靠。

设置两个index，left记录第一个1的位置，left左边为0，right记录第一个非2的位置，right右边为2.

然后使用i从头到尾扫一遍，直到与right相遇。

i遇到0就换到左边去，遇到2就换到右边去，遇到1就跳过。

需要注意的是：由于left记录第一个1的位置，因此A[left]与A[i]交换后，A[left]为0,A[i]为1，因此i++；

而right记录第一个非2的位置，可能为0或1，因此A[right]与A[i]交换后，A[right]为2,A[i]为0或1，i不能前进，要后续判断。

由此该数组分为4段：[0,left)-->0; [left,i)-->1; [i,right]-->乱序; (right,n-1]-->2

0 0 0 1 1 1 2 1 0 2 1 2 2 2

^ ^ ^

left i right

way-3:【最直观：平移插入】

\*/

class Solution {

public:

void sortColors(vector<int>& nums)

{

//way-1

/\*

int zero=0;

int one=0;

int two=0;

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

{

if(nums[i]==0)

zero++;

else if(nums[i]==1)

one++;

else

two++;

}

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

{

if(zero!=0)

{

nums[i]=0;

zero--;

}

else if(one!=0)

{

nums[i]=1;

one--;

}

else

{

nums[i]=2;

}

}

return;

\*/

//way-2

int left = 0;

int right = n-1;

int i = 0;

while(i <= right)

{

if(A[i] == 0)

{

swap(A[left], A[i]);

left ++;

i ++;

}

else if(A[i] == 1)

{

i ++;

}

else

{

swap(A[i], A[right]);

right --;

}

}

//way-3

/\*

int i = -1;

int j = -1;

int k = -1;

int n=nums.size();

for(int p = 0; p < n; p ++)

{

//根据第i个数字，挪动0~i-1串。

if(nums[p] == 0)

{

nums[++k] = 2; //2往后挪

nums[++j] = 1; //1往后挪

nums[++i] = 0; //0往后挪

}

else if(nums[p] == 1)

{

nums[++k] = 2;

nums[++j] = 1;

}

else

nums[++k] = 2;

}

\*/

}

};