Given an array and a value, remove all instances of that value [**in-place**](https://en.wikipedia.org/wiki/In-place_algorithm) and return the new length.

Do not allocate extra space for another array, you must do this by **modifying the input array**[**in-place**](https://en.wikipedia.org/wiki/In-place_algorithm) with O(1) extra memory.

The order of elements can be changed. It doesn't matter what you leave beyond the new length.

**Example:**

Given **nums** = [3,2,2,3], **val** = 3,

Your function should return length = 2, with the first two elements of *nums* being 2.

题目：删除数组中的val值，并返回剩下数组的长度。

注意：这里题目要求删除数组的某个值，这里并不用真的做到删除，如果你想用后边的值覆盖，最后的元素值也是消除不掉的，所以在以后做到要删除什么值的时候就直接把该值丢到数组最后就好了。还有一种思路就是把数组后边不等于要删除的数值覆盖掉要删除的。覆盖掉一个，数组的记录长度就要-1。数组当前长度之后的数据等于多少都是废弃的。

我的想法，从头遍历数组，然后把等于val的数据和从尾巴开始不等于val的数据互换。这样做可以把所有的等于val的数据都换到最后，数组中所有的数据都不会丢失，只不过把所有等于val的数据都换到的数组的最后。

代码如下：

public static int removeElement(int[] nums, int val) {

int i = nums.length-1;

for(int j=0;j<i;j++){ // 注意这里的i为当前数组的长度，而不是nums.length

if(nums[j]==val){

while (nums[i]==val){

i--;

}

int temp = nums[j];

nums[j] = nums[i];

nums[i] = temp;

}

}

return i;

}

注意的是：当一个数据换到数组后边，那么数据长度应该-1，后边的数据就当作被剪掉，如果还保留的话会造成重复的错误。

答案，跟我不同的是：我的想法是把重复的数据换到数组后边，这样数据还都完好保存只不过是换了位置。答案是用后边不等于val的数据替换掉前边等于val的数据。因为val是要删除的数据，并不用考虑数组数据的完整性，直接覆盖掉就可以了，并不用费力去替换位置。

答案用两个指针，i和j。i和j都是从头开始，i当当前数据不等于val才会向下走一步，j无条件遍历整个数组。j的速度一定大于或等于i的速度。用j遍历整个数组，找到不等于val的数据。i到等于val的数据会停下。当i的速度等于j的速度，也就是说没有遇到等于val的数据，i一直往下走，j也往下走，同时i==j，nums[j]！=val,会执行nums[i]=nums[j]，但是i=j，nums[i]和nums[j]为同一个数据单元，nums[i]=nums[j]并没有覆盖掉不等于val的i数据。当num[j]==val的时候，也就是说j遇到等于val的数据的时候，i会停下，然后到j遇到下一个不等于val的数据，然后将那个不等于val的数据nums[j]把前面等于val的数据nums[i]覆盖掉，然后i才到下一步。当j走完一遍，就是遍历完一遍数组所有的等于val的数据也会被找到，会被i记录，然后由下一个不等于val的数据替换掉，i继续走，其实i的长度就是不等于val的数据的长度。

代码：

public static int removeElement\_A(int[] nums,int val){

int i = 0;

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

if(nums[j]!=val){

nums[i]=nums[j];

i++;

}

}

return i;

}

这个算法简单来说就是，当遇到等于val的数据的时候i会停下，j不会停下。j-i为等于val的数据个数，当遇到不等val的元素会前移j-i个单元，就是用nums[j]覆盖掉nums[i]。同时向前覆盖的那几个元素也是重复的，本身位置没有变，前边等于val的元素数值改为后边的不等于val的元素数值，后边的也就是原本的数据就是重复的，覆盖掉数据也不会丢。也就是说每个元素都移动j-i个数据单元。i始终是不等于val的数据单元个数，j为当前位置由有多少数据单元，j-i始终是数组中val的个数。当nums[j]为当前数据单元，前边有几个等于val的数据单元就向前移动几个单元，目标单元的位置就是nums[i]。

体会：1.要删除数组的元素，直接用后边不用删除的元素覆盖掉即可

2.把不要的元素直接丢到数组尾巴，不要费事的把所有后边元素前移。

3.这些题目是不要求删除后元素保持原来次序的。要保持原来次序所有元素都要前移才对。