We have some permutation A of [0, 1, ..., N - 1], where N is the length of A.

The number of (global) inversions is the number of i < j with 0 <= i < j < N and A[i] > A[j].

The number of local inversions is the number of i with 0 <= i < N and A[i] > A[i+1].

Return true if and only if the number of global inversions is equal to the number of local inversions.

**Example 1:**

**Input:** A = [1,0,2]

**Output:** true

**Explanation:** There is 1 global inversion, and 1 local inversion.

**Example 2:**

**Input:** A = [1,2,0]

**Output:** false

**Explanation:** There are 2 global inversions, and 1 local inversion.

**Note:**

* A will be a permutation of [0, 1, ..., A.length - 1].
* A will have length in range [1, 5000].
* The time limit for this problem has been reduced.

题目

当A[i]>A[i+1],此时为一次局部反演

当i<j,A[i]>A[j],此时为一次全局反演

首先，你要明白局部反演是 全局反演的一种特殊形式，就是当j=i+1的时候。然后我们看题，题目说原来初始序列是[0,1,2,3,…,n]这种递增数列，然后进行变化得到的。我们看例子，

[1,0,2],全局反演等于局部反演，我们仔细观察，这个数列只是0和1对换得到的，也就是说，1应该在位置1，现在在位置0，1向前偏移了一位。同时出现的反演是局部反演，同时也属于全局反演。我们再看[1,2,0]，你会发现原来0在位置0，现在出现在了位置2.这时候产生的反演是全局反演，但是可不是局部反演，因为它向前偏移了两位，这时候全局反演次数大于局部反演次数。

我们要直到到，当一个数字偏移位置的绝对值小于等于1的时候，属于局部反演，也是全局反演。因为原本序列是有序的。当一个数字偏移位置的绝对值超过1，就说明它不能满足了i和i+1的反演关系了，同时这次出现的反演是全局反演不是局部反演。

所以我们需要找位置的偏移量就好了。

题目告诉我们原来序列是[0,1,2,…,n]，0在0的位置，1在1的位置，A[i]应该在i的位置,A[i]-i为数字的偏移量。当A[i]-i>1就说明全局反演大于了局部反演，返回false。否则全部反演一直等于局部反演，返回true。

代码：

for(int i = 0;i<A.length;i++){

if(Math.abs(A[i]-i)>1)

return false;

}

return true;

下面是我做的从头和尾巴同时开始计算各个反演的数量，原因是没有体会到题目的真正精髓。

public static boolean isIdealPermutation(int[] A) {

int i = 0;

int j = A.length-2;

int global = 0;

int local =0;

while(i<=j){

for(int m = i+1;m<A.length;m++){

if(A[i]>A[m]){

global++;

if(m==i+1)

local++;

}

}

if(i==j)

break;

i++;

for(int m = j+1;m<A.length;m++){

if(A[j]>A[m]){

global++;

if(m==j+1)

local++;

}

}

j--;

}

return global==local;

}