**描述**

给定一个**旋转**排序数组，在原地恢复其排序。（升序）

**样例**

**Example1:**  
[4, 5, 1, 2, 3] -> [1, 2, 3, 4, 5]  
**Example2:**  
[6,8,9,1,2] -> [1,2,6,8,9]

**挑战**

使用O(1)的额外空间和O(*n*)时间复杂度

我对旋转排序数组一开始的想法是能不能自己搞一个旋转排序数组，然后我有了旋转排序数组的思路后我再回头搞一下这道题。但是后来又看了看它说的挑战，第一个想法是直接归并排序。但是归并排序的话要辅助数组，就不符合题意了。然后归并排序不行的话，那完全可以上快速排序嘛。反正快排是绝对满足这个要求的。然后我就用快排把这道题写出来了。

代码如下：

void exchange(vector<int>& a, const int& i, const int& j) {

int tmp = a[i];

a[i] = a[j];

a[j] = tmp;

}

int partition(vector<int>& a,const int &lo,const int &hi) {

int i = lo, j = hi + 1;

int val = a[lo];

while (true) {

while (a[++i] < val)

if (i == hi)

break;

while (val < a[--j])

if (j == lo)

break;

if (i >= j)

break;

exchange(a, i, j);

}

exchange(a, lo, j);

return j;

}

void quicksort(vector<int>& a,const int&lo,const int&hi) {

if (lo >= hi)

return;

int j = partition(a, lo, hi);

quicksort(a, lo, j - 1);

quicksort(a, j + 1, hi);

}

void sortofquick(vector<int>& a) {

int n = a.size() - 1;

quicksort(a, 0, n);

}

void recoverRotatedSortedArray(vector<int> &nums) {

// write your code here

sortofquick(nums);

}

截图如下：

