**[aston](https://www.cnblogs.com/aston/)**

## [八种排序算法（内部排序）](https://www.cnblogs.com/aston/p/5894146.html)

　　八种排序算法很长时间没有使用了，今天做一个总结，方便以后自己用的时候参考。

　　这八种排序算法都是内部算法，这八种排序算法分别是：

　　1. 插入排序  
　　　　1）直接插入排序  
　　　　2）希尔排序  
　　2.选择排序  
　　　　1）简单选择排序  
　　　　2）堆排序  
　　3.交换排序  
　　　　1）冒泡排序  
　　　　2）快速排序  
　　4.归并排序  
　　5.基数排序

一、直接插入排序

　　将一个记录插入到已经排好序的有序表中，从而得到一个新的、记录数增1的有序表。在实际操作中，先将序列的第一个记录看成是一个有序的子序列，然后从第二个、第三个、……记录逐个进行插入，直至整个序列有序为止。

　　插入排序算法Java实现：

　　public class DirectInsertionSort {  
　　　　/\*\*  
　　　　\* 直接插入排序  
　　　　\* @param args  
　　　　\*/  
　　　　public static void main(String[] args) {  
　　　　　　int[] arrayInt = {17,56,80,17,12,9,100,64,42,37,64,82,123,974,314,548};  
　　　　　　int length = arrayInt.length;  
　　　　　　  
　　　　　　for( int i=1; i<length; i++){  
　　　　　　　　for( int j=i; j>0; j--){  
　　　　　　　　　　if( arrayInt[j] <= arrayInt[j-1]){  
　　　　　　　　　　　　int k = arrayInt[j];  
　　　　　　　　　　　　arrayInt[j] = arrayInt[j-1];  
　　　　　　　　　　　　arrayInt[j-1] = k;  
　　　　　　　　　　}  
　　　　　　　　}  
　　　　　　}  
  
　　　　　　for( int i=0; i<length; i++){  
　　　　　　　　System.out.println( arrayInt[i]);  
　　　　　　}  
　　　　}

　　}

二、希尔排序

　　先将整个待排序的记录序列分割成为若干子序列分别进行直接插入排序，带整个序列中的记录“基本有序”时，再对全体记录进行依次直接插入排序。

　　希尔排序Java实现：

　　public class ShellSort {  
　　　　/\*\*  
　　　　\* 希尔排序  
　　　　\* @param args  
　　　　\*/  
　　　　public static void main(String[] args) {  
　　　　　　int[] arrayInt = {17,56,80,17,12,9,100,64,42,37,64,82,123,974,314,548};  
　　　　　　int length = arrayInt.length;  
　　　　　　System.out.println( "length="+length);  
  
　　　　　　shellSort( arrayInt);  
  
　　　　　　for( int i=0; i<length; i++){  
　　　　　　　　System.out.println( "arrayInt[" + i + "]=" + arrayInt[i]);  
　　　　　　}

　　　　}  
  
　　　　public static void shellSort( int[] arrayInt){  
　　　　　　int j = 0;  
　　　　　　int temp = 0;  
　　　　　　for ( int increment=arrayInt.length/2; increment>0; increment/=2){  
　　　　　　　　for ( int i=increment; i<arrayInt.length; i++){  
　　　　　　　　　　temp = arrayInt[i];  
　　　　　　　　　　for ( j=i; j>=increment; j-=increment){  
　　　　　　　　　　　　if( temp > arrayInt[j-increment]){  
　　　　　　　　　　　　　　arrayInt[j] = arrayInt[j - increment];  
　　　　　　　　　　　　}else{  
　　　　　　　　　　　　　　break;  
　　　　　　　　　　　　}  
　　　　　　　　　　}  
　　　　　　　　　　arrayInt[j] = temp;  
　　　　　　　　}  
　　　　　　}  
　　　　}

　　}

三、简单选择排序

　　在待排序的一组数中，选出最小（大）的一个数与第一个位置的数交换；然后在剩下的数当中再找最小（大）的与第二个位置的数交换，以此类推，直到第n-1个元素和第n个元素比较为止。

　　简单选择排序Java实现：

　　public class SimpleSelectionSort {  
　　　　/\*\*  
　　　　\* 简单选择排序  
　　　　\* @param args  
　　　　\*/  
　　　　public static void main(String[] args) {  
　　　　　　int[] arrayInt = {17,56,80,17,12,9,100,64,42,37,64,82,123,974,314,548};  
　　　　　　int length = arrayInt.length;  
  
　　　　　　for( int i=0; i<length; i++){  
　　　　　　　　for( int j=i+1; j<length; j++){  
　　　　　　　　　　if( arrayInt[i] > arrayInt[j]){  
　　　　　　　　　　　　int k = arrayInt[i];  
　　　　　　　　　　　　arrayInt[i] = arrayInt[j];  
　　　　　　　　　　　　arrayInt[j] = k;  
　　　　　　　　　　}  
　　　　　　　　}  
　　　　　　}  
　　　　　　  
　　　　　　for( int i=0; i<length; i++){  
　　　　　　　　System.out.println( arrayInt[i]);  
　　　　　　}  
　　　　}

　　}

四、堆排序

　　一种树形选择排序，是对直接选择排序的有效改进。

　　堆排序Java实现：

五、冒泡排序

　　对待排序的一组数，自上而下对相邻的两个数依次进行比较和调整，让较大（小）的数往下沉，较小（大）的往上冒。即：每当两相邻的数比较后发现它们的排序与排序要求相反时，就将它们互换。

　　冒泡排序Java实现：

　　public class BubbleSort {  
　　　　/\*\*  
　　　　\* 冒泡排序  
　　　　\* @param args  
　　　　\*/  
　　　　public static void main(String[] args) {  
　　　　　　int[] arrayInt = {17,56,80,17,12,9,100,64,42,37,64,82,123,974,314,548};  
　　　　　　int length = arrayInt.length;  
　　　　　　  
　　　　　　for( int i=0; i<length-1; i++){  
　　　　　　　　for( int j=0; j<length-i-1; j++){  
　　　　　　　　　　if( arrayInt[j] > arrayInt[j+1]){  
　　　　　　　　　　　　int k = arrayInt[j];  
　　　　　　　　　　　　arrayInt[j] = arrayInt[j+1];  
　　　　　　　　　　　　arrayInt[j+1] = k;  
　　　　　　　　　　}  
　　　　　　　　}  
　　　　　　}  
　　　　　　  
　　　　　　for( int i=0; i<length; i++){  
　　　　　　　　System.out.println( arrayInt[i]);  
　　　　　　}  
　　　　}

　　}

六、快速排序

　　通过一趟排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一部分的所有数据都要小，然后再按此方法对这两部分数据分别进行快速排序，整个排序过程可以递归进行，以此达到整个数据变成有序序列。

　　快速排序Java实现：

　　public class QuickSort {  
　　　　/\*\*  
　　　　\* 快速排序  
　　　　\* @param args  
　　　　\*/  
　　　　public static void main(String[] args) {  
　　　　　　int[] arrayInt = {17,56,80,17,12,9,100,64,42,37,64,82,123,974,314,548};  
　　　　　　int start = 0;  
　　　　　　int end = arrayInt.length-1;  
　　　　　　  
　　　　　　quickSort( arrayInt, start, end);  
　　　　　　  
　　　　}  
  
　　　　public static void quickSort( int[] arrayInt, int start, int end){  
　　　　　　if( start >= end){  
　　　　　　　　return;  
　　　　　　}  
　　　　　　  
　　　　　　int i = start;  
　　　　　　int j = end;  
　　　　　　int value = arrayInt[start];  
　　　　　　boolean flag = true;  
　　　　　　  
　　　　　　while( i != j){  
　　　　　　　　if( flag){  
　　　　　　　　　　if( value > arrayInt[j]){  
　　　　　　　　　　　　int k = arrayInt[i];  
　　　　　　　　　　　　arrayInt[i] = arrayInt[j];  
　　　　　　　　　　　　arrayInt[j] = k;  
　　　　　　　　　　　　i++;  
　　　　　　　　　　　　flag = false;  
　　　　　　　　　　}else{  
　　　　　　　　　　　　j--;  
　　　　　　　　　　}  
　　　　　　　　}else{  
　　　　　　　　　　if( value < arrayInt[i]){  
　　　　　　　　　　　　int k = arrayInt[i];  
　　　　　　　　　　　　arrayInt[i] = arrayInt[j];  
　　　　　　　　　　　　arrayInt[j] = k;  
　　　　　　　　　　　　j--;  
　　　　　　　　　　　　flag = true;  
　　　　　　　　　　}else{  
　　　　　　　　　　　　i++;  
　　　　　　　　　　}  
　　　　　　　　}  
　　　　　　}  
　　　　　　  
　　　　　　for( int m=0; m<arrayInt.length; m++){  
　　　　　　　　System.out.print(arrayInt[m] + " ");  
　　　　　　}  
　　　　　　System.out.println();  
　　　　　　  
　　　　　　quickSort( arrayInt, start, j-1);  
　　　　　　quickSort( arrayInt, i+1, end);  
　　　　}  
　　　　  
　　}

七、归并排序

　　建立在归并操作上的一种有效的排序算法，该算法是采用“分治法”的一个非常典型的应用。

　　归并排序Java实现：

　　public class MergeSort {  
　　　　/\*\*  
　　　　\* 归并排序  
　　　　\* @param args  
　　　　\*/  
　　　　public static void main(String[] args) {  
　　　　　　int[] arrayInt = {17,56,80,17,12,9,100,64,42,37,64,82,123,974,314,548};  
　　　　　　int start = 0;  
　　　　　　int end = arrayInt.length-1;  
　　　　　　  
　　　　　　mergeSort( arrayInt, start, end);  
　　　　　　  
　　　　　　for( int i=0; i<arrayInt.length; i++){  
　　　　　　　　System.out.print( arrayInt[i]+" ");  
　　　　　　}  
　　　　}  
  
　　　　public static int[] mergeSort( int[] arrayInt, int start, int end){  
　　　　　　int middle = ( start+end)/2;  
　　　　　　  
　　　　　　if( start < end){  
　　　　　　　　mergeSort( arrayInt, start, middle);  
　　　　　　　　mergeSort( arrayInt, middle+1, end);  
　　　　　　　　sort( arrayInt, start, middle, end);  
　　　　　　}  
　　　　　　  
　　　　　　return arrayInt;  
　　　　}  
　　　　  
　　　　public static void sort( int[] arrayInt, int start, int middle, int end){  
　　　　　　int[] arrayTemp = new int[ end-start+1];  
　　　　　　int i = start;  
　　　　　　int j = middle +1;  
　　　　　　int k = 0;  
　　　　　　  
　　　　　　while( i <= middle && j <= end){  
　　　　　　　　if( arrayInt[i] < arrayInt[j]){  
　　　　　　　　　　arrayTemp[k] = arrayInt[i];  
　　　　　　　　　　k++;  
　　　　　　　　　　i++;  
　　　　　　　　}else{  
　　　　　　　　　　arrayTemp[k] = arrayInt[j];  
　　　　　　　　　　k++;  
　　　　　　　　　　j++;  
　　　　　　　　}  
　　　　　　}  
　　　　　　  
　　　　　　while( i<=middle){  
　　　　　　　　arrayTemp[k] = arrayInt[i];  
　　　　　　　　k++;  
　　　　　　　　i++;  
　　　　　　}  
　　　　　　  
　　　　　　while( j <= end){  
　　　　　　　　arrayTemp[k] = arrayInt[j];  
　　　　　　　　k++;  
　　　　　　　　j++;  
　　　　　　}  
　　　　　　  
　　　　　　for( int m=0; m<arrayTemp.length; m++){  
　　　　　　　　arrayInt[m+start] = arrayTemp[m];  
　　　　　　}  
　　　　}  
  
　　}

八、基数排序

　　属于“分配式排序”，又称“桶子法”。它是透过键值的部份资讯，将要排序的元素分配至某些“桶”中，藉以达到排序的作用。基数排序法是属于稳定性的排序。

　　基数排序Java排序：



# **快速排序（过程图解）**



[Mr\_249](https://blog.csdn.net/pengzonglu7292" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank) 2018-12-10 14:55:38 IMG_257 36638 IMG_258 收藏 98

分类专栏： [Java](https://blog.csdn.net/pengzonglu7292/category_8298475.html" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank) [java从入门到精通](https://blog.csdn.net/pengzonglu7292/category_9286994.html" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank) [数据结构与算法](https://blog.csdn.net/pengzonglu7292/category_9287051.html" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)

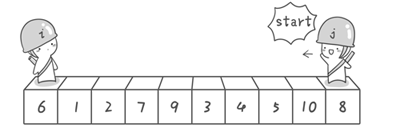
假设我们现在对“6  1  2 7  9  3  4  5 10  8”这个10个数进行排序。首先在这个序列中随便找一个数作为基准数（不要被这个名词吓到了，就是一个用来参照的数，待会你就知道它用来做啥的了）。为了方便，就让第一个数6作为基准数吧。接下来，需要将这个序列中所有比基准数大的数放在6的右边，比基准数小的数放在6的左边，类似下面这种排列。

       3  1  2 5  4  6  9 7  10  8

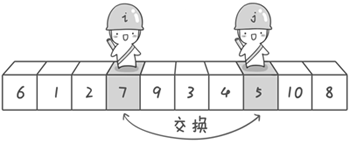
        在初始状态下，数字6在序列的第1位。我们的目标是将6挪到序列中间的某个位置，假设这个位置是k。现在就需要寻找这个k，并且以第k位为分界点，左边的数都小于等于6，右边的数都大于等于6。想一想，你有办法可以做到这点吗？

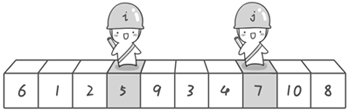
        给你一个提示吧。请回忆一下冒泡排序，是如何通过“交换”，一步步让每个数归位的。此时你也可以通过“交换”的方法来达到目的。具体是如何一步步交换呢？怎样交换才既方便又节省时间呢？先别急着往下看，拿出笔来，在纸上画画看。我高中时第一次学习冒泡排序算法的时候，就觉得冒泡排序很浪费时间，每次都只能对相邻的两个数进行比较，这显然太不合理了。于是我就想了一个办法，后来才知道原来这就是“快速排序”，请允许我小小的自恋一下(^o^)。

        方法其实很简单：分别从初始序列“6  1  2 7  9  3  4  5 10  8”两端开始“探测”。先从右往左找一个小于6的数，再从左往右找一个大于6的数，然后交换他们。这里可以用两个变量i和j，分别指向序列最左边和最右边。我们为这两个变量起个好听的名字“哨兵i”和“哨兵j”。刚开始的时候让哨兵i指向序列的最左边（即i=1），指向数字6。让哨兵j指向序列的最右边（即j=10），指向数字8。



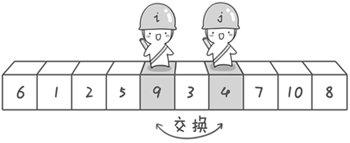
首先哨兵j开始出动。因为此处设置的基准数是最左边的数，所以需要让哨兵j先出动，这一点非常重要（请自己想一想为什么）。哨兵j一步一步地向左挪动（即j--），直到找到一个小于6的数停下来。接下来哨兵i再一步一步向右挪动（即i++），直到找到一个数大于6的数停下来。最后哨兵j停在了数字5面前，哨兵i停在了数字7面前。

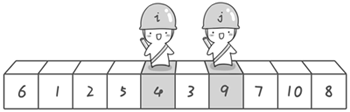




现在交换哨兵i和哨兵j所指向的元素的值。交换之后的序列如下。

        6  1  2  5  9 3  4  7  10  8



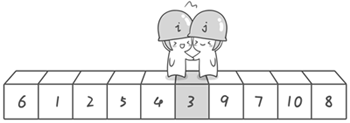


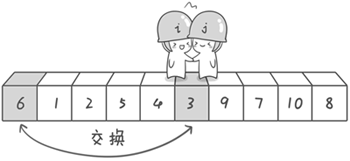
  到此，第一次交换结束。接下来开始哨兵j继续向左挪动（再友情提醒，每次必须是哨兵j先出发）。他发现了4（比基准数6要小，满足要求）之后停了下来。哨兵i也继续向右挪动的，他发现了9（比基准数6要大，满足要求）之后停了下来。此时再次进行交换，交换之后的序列如下。

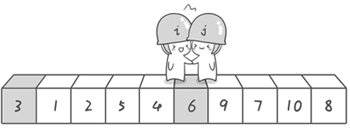
        6  1  2 5  4  3  9  7 10  8

        第二次交换结束，“探测”继续。哨兵j继续向左挪动，他发现了3（比基准数6要小，满足要求）之后又停了下来。哨兵i继续向右移动，糟啦！此时哨兵i和哨兵j相遇了，哨兵i和哨兵j都走到3面前。说明此时“探测”结束。我们将基准数6和3进行交换。交换之后的序列如下。

        3  1 2  5  4  6  9 7  10  8







到此第一轮“探测”真正结束。此时以基准数6为分界点，6左边的数都小于等于6，6右边的数都大于等于6。回顾一下刚才的过程，其实哨兵j的使命就是要找小于基准数的数，而哨兵i的使命就是要找大于基准数的数，直到i和j碰头为止。

        OK，解释完毕。现在基准数6已经归位，它正好处在序列的第6位。此时我们已经将原来的序列，以6为分界点拆分成了两个序列，左边的序列是“3  1 2  5  4”，右边的序列是“9  7  10  8”。接下来还需要分别处理这两个序列。因为6左边和右边的序列目前都还是很混乱的。不过不要紧，我们已经掌握了方法，接下来只要模拟刚才的方法分别处理6左边和右边的序列即可。现在先来处理6左边的序列现吧。

        左边的序列是“3  1  2 5  4”。请将这个序列以3为基准数进行调整，使得3左边的数都小于等于3，3右边的数都大于等于3。好了开始动笔吧。

        如果你模拟的没有错，调整完毕之后的序列的顺序应该是。

        2  1  3  5  4

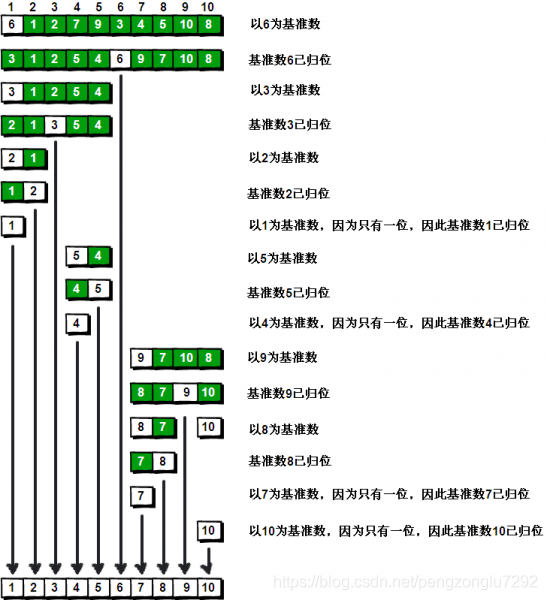
        OK，现在3已经归位。接下来需要处理3左边的序列“2 1”和右边的序列“5 4”。对序列“2 1”以2为基准数进行调整，处理完毕之后的序列为“1 2”，到此2已经归位。序列“1”只有一个数，也不需要进行任何处理。至此我们对序列“2 1”已全部处理完毕，得到序列是“1 2”。序列“5 4”的处理也仿照此方法，最后得到的序列如下。

        1  2  3 4  5  6 9  7  10  8

        对于序列“9  7  10  8”也模拟刚才的过程，直到不可拆分出新的子序列为止。最终将会得到这样的序列，如下。

        1  2  3 4  5  6  7  8 9  10

        到此，排序完全结束。细心的同学可能已经发现，快速排序的每一轮处理其实就是将这一轮的基准数归位，直到所有的数都归位为止，排序就结束了。下面上个霸气的图来描述下整个算法的处理过程。



  快速排序之所比较快，因为相比冒泡排序，每次交换是跳跃式的。每次排序的时候设置一个基准点，将小于等于基准点的数全部放到基准点的左边，将大于等于基准点的数全部放到基准点的右边。这样在每次交换的时候就不会像冒泡排序一样每次只能在相邻的数之间进行交换，交换的距离就大的多了。因此总的比较和交换次数就少了，速度自然就提高了。当然在最坏的情况下，仍可能是相邻的两个数进行了交换。因此快速排序的最差时间复杂度和冒泡排序是一样的都是O(N2)，它的平均时间复杂度为O(NlogN)。

#include <stdio.h>

int a[101],n;*//定义全局变量，这两个变量需要在子函数中使用*

void quicksort(int left, int right) {

int i, j, t, temp;

if(left > right)

return;

temp = a[left]; *//temp中存的就是基准数*

i = left;

j = right;

while(i != j) { *//顺序很重要，要先从右边开始找*

while(a[j] >= temp && i < j)

j--;

while(a[i] <= temp && i < j)*//再找右边的*

i++;

if(i < j)*//交换两个数在数组中的位置*

{

t = a[i];

a[i] = a[j];

a[j] = t;

}

}

*//最终将基准数归位*

a[left] = a[i];

a[i] = temp;

quicksort(left, i-1);*//继续处理左边的，这里是一个递归的过程*

quicksort(i+1, right);*//继续处理右边的 ，这里是一个递归的过程*

}

int main() {

int i;

*//读入数据*

scanf("%d", &n);

for(i = 1; i <= n; i++)

scanf("%d", &a[i]);

quicksort(1, n); *//快速排序调用*

*//输出排序后的结果*

for(i = 1; i < n; i++)

printf("%d ", a[i]);

printf("%d\n", a[n]);

return 0;

}

原文链接：

<http://www.cnblogs.com/ahalei/p/3568434.html>

* IMG_268点赞96
* [IMG_269评论29](https://blog.csdn.net/pengzonglu7292/article/details/84938910" \l "commentBox)
* [IMG_270分享](https://blog.csdn.net/pengzonglu7292/article/details/javascript:;)
* [IMG_271收藏98](https://blog.csdn.net/pengzonglu7292/article/details/javascript:;)
* IMG_272举报
* [关注](https://blog.csdn.net/pengzonglu7292/article/details/javascript:;)

一键三连

[快速排序法](https://blog.csdn.net/u014600626/article/details/50876346" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)

[gcs的博客](https://blog.csdn.net/u014600626" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)

IMG_273 2543

[今天介绍快速排序，这也是在实际中最常用的一种排序算法，速度快，效率高。就像名字一样，快速排序是最优秀的一种排序算法。 思想 快速排序采用的思想是分治思想。 快速排序是找出一个元素（理论上可以随便找一个）作为基准(pivot),然后对数组进行分区操作,使基准左边元素的值都不大于基准值,基准右边的元素值 都不小于基准值，如此作为基准的元素调整到排序后的正确位置。递归快速排序，将其他n](https://blog.csdn.net/u014600626/article/details/50876346" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)

[快速排序图解及实现](https://blog.csdn.net/qq_16635277/article/details/86572416" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)

[nrxsh](https://blog.csdn.net/qq_16635277" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)

IMG_274 3122

[用C语言实现快速排序](https://blog.csdn.net/qq_16635277/article/details/86572416" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)

[](https://blog.csdn.net/xiaohe6688)

窗体顶端

IMG_276

窗体底端

[](https://blog.csdn.net/weixin_43218580)

[weixin\_43218580](https://blog.csdn.net/weixin_43218580" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)**:**错的，正确表达应该是赋值，而不是交换。如果用这种方法去排其他的序，很明显是错的7月前回复IMG_278

点赞3

[](https://blog.csdn.net/weixin_43863672)

[小Ana](https://blog.csdn.net/weixin_43863672" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)回复weixin\_45740217**:**这个好像就是错的，反正就是不太用这个方法了。一般的方法是将6先保存一下，然后数组6这个位置看成是空的，然后从后往前找，如果遇到大于6的，就将值赋给6这个位置，然后原来大于6的位置变成空的，再从头往后找。。。最后把6拿过来放在空的位置上。6月前回复IMG_281

点赞1

[](https://blog.csdn.net/weixin_45740217)

[weixin\_45740217](https://blog.csdn.net/weixin_45740217" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)回复**:**没太听懂，具体错在哪里，哪一行可以说明一下吗？7月前回复IMG_284

点赞

[IMG_286](https://blog.csdn.net/xclltssun)

[小橙子](https://blog.csdn.net/xclltssun" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)**:**6 1 2 7 9 11 4 5 10 8 如果中间不是3是11怎么办？6和11交换？9月前回复IMG_287

点赞1

[](https://blog.csdn.net/weixin_45740217)

[weixin\_45740217](https://blog.csdn.net/weixin_45740217" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)回复骑车车**:**这个“小橙子”它把这一串数字写的是最原始的数据，正确的数字应该是6 1 2 5 4 11 9 7 10 8，这里的11原本是3，原本哨兵i和j走到3就停下来了，他想问的是，如果将3换成了11会怎么办。7月前回复IMG_290

点赞

[](https://blog.csdn.net/qq_41862978)

[骑车车](https://blog.csdn.net/qq_41862978" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)回复weixin\_45740217**:**4和6交换后，4的左边还是有5，是不是就错了7月前回复IMG_293

点赞

[](https://blog.csdn.net/qq_41862978)

[骑车车](https://blog.csdn.net/qq_41862978" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)回复weixin\_45740217**:**那也不对呀，推算结果后是6和4换了位置，但是4左边还有一个5比它大，是不是就错了呢7月前回复IMG_296

点赞

[](https://blog.csdn.net/weixin_45740217)

[weixin\_45740217](https://blog.csdn.net/weixin_45740217" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)回复**:**士兵j遇见3停下来是因为3小于了基准数6，如果遇见了大于基准数6的11则不会停下来继续往左边走，直至遇见小于基准数6的4，最后再将6和士兵i和j会面的4交换位置。7月前回复IMG_299

点赞

[](https://blog.csdn.net/Rookie_______)

[@LXH](https://blog.csdn.net/Rookie_______" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)回复**:**士兵j先走，会在4那个位置士兵i，和j相遇8月前回复IMG_302

点赞

[](https://blog.csdn.net/weixin_45479946)

[yoka\_milk](https://blog.csdn.net/weixin_45479946" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)**:**搞清楚了，首先搞清一个原则，最后与基准数的交换，一定是与一个小于基准数的数交换！但是链接错误的地方在于，它的结束条件是两边指针汇合处，但汇合处可能大于0，也可能小于0，所以结束条件必须是左边指针大于右边指针，然后最后交换只能是和j交换，因为j暂停的时候指向是小于基准数的数，i大于j的时候，j指向也一定是小于基准数的数，这样最后交换才会成立。所以不论哪个指针先动都可以，只要改变结束条件就行！后面有我参考修正后的代码！\*\*3月前回复IMG_305

点赞

[](https://blog.csdn.net/weixin_45479946)

[yoka\_milk](https://blog.csdn.net/weixin_45479946" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)**:**\*我自己试了试，发现，如果i先开始移动，就会导致最后会合的地方在大于基数的数，这样就没办法交换了，而如果j先开始移动，最后一定是在小于基数的地方停下，这样一交换，小于基数的数就跑到前面去了！3月前回复IMG_308

点赞

[](https://blog.csdn.net/weixin_45714185)

[weixin\_45714185](https://blog.csdn.net/weixin_45714185" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)**:**为什么必须右边先开始，从从左边先开始，就错了，换成右边就对了，谁给解释下4月前回复IMG_311

点赞

[](https://blog.csdn.net/weixin_45479946)

[yoka\_milk](https://blog.csdn.net/weixin_45479946" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)回复**:**因为基准数一直定在左边。我试过，左右两边都可以开始，如果从右边开始，最后基准数与左边的指针i交换，如果从左边开始，基准数最后与j交换。1月前回复IMG_314

点赞

[](https://blog.csdn.net/Mr_Qian_Ives)

[码哥码哥](https://blog.csdn.net/blogdevteam/article/details/103478461" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)[钱甫新](https://blog.csdn.net/Mr_Qian_Ives" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)回复**:**你设置的比较值是左边第一个，所以从右边开始1月前回复IMG_318

点赞

[](https://blog.csdn.net/u014183815)

[我要学好数据结构](https://blog.csdn.net/u014183815" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)**:**如何将5作为基准数？5并不是开头的数字5月前回复IMG_321

点赞

[](https://blog.csdn.net/u014183815)

[我要学好数据结构](https://blog.csdn.net/u014183815" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)**:**请问为什么left>right的时候就要return呢？5月前回复IMG_324

点赞

[](https://blog.csdn.net/u014183815)

[我要学好数据结构](https://blog.csdn.net/u014183815" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)**:**大佬画图用的什么软件能透露下吗？5月前回复IMG_327

点赞

[](https://blog.csdn.net/u014183815)

[我要学好数据结构](https://blog.csdn.net/u014183815" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)**:**请问你这个哨兵怎么画上去的？用的什么软件呢？5月前回复IMG_330

点赞

[](https://blog.csdn.net/weixin_40480741)

[跨海之梦](https://blog.csdn.net/weixin_40480741" \t "https://blog.csdn.net/pengzonglu7292/article/details/_blank)**:**为你点赞 秒懂6月前回复IMG_333

点赞

* <
* **1**
* 2
* >