**快排quickSort即相关题型的解答**[](#quicksort)

使用到快排技术的题目： 比如荷兰国旗问题，求前k个最小值问题，但是所有的快排问题需要解决经典快排的缺陷

面试题40. 最小的k个数

**class** **Solution** **{**

**public** **int[]** **getLeastNumbers(int[]** arr**,** **int** k**)** **{**

**if(**k**==**0**||**arr**.**length**==**0**)** **return** **new** **int[**0**];**

partitionArr**(**0**,**arr**.**length**-**1**,**arr**,**k**-**1**);**

**int[]** res **=** **new** **int[**k**];**

**for(int** i**=**0**;**i**<**k**;**i**++){**

res**[**i**]** **=** arr**[**i**];**

**}**

**return** res**;**

**}**

**public** **void** **partitionArr(int** L**,int** R**,int[]** arr**,** **int** k**){**

**int** m **=** partition**(**L**,**R**,**arr**,**k**);**

**if(**m**>**k**){**

partitionArr**(**L**,**m**-**1**,**arr**,**k**);**

**}else** **if(**m**<**k**){**

*//partitionArr(L,m+1,arr,k-m); 容易出错的地方*

partitionArr**(**m**+**1**,**R**,**arr**,**k**);**

**}else{**

**return;**

**}**

**}**

*/\*\**

*\* 快排的实现思路：*

*\* 1.先取一个基准位置的值val,经典快排都是取数组最右位置的值val=arr[R]*

*\* 2.初始化小于区域less=L-1, 初始化大于区域为more=R(这里要主要注意为什么不是more= R+1，在荷兰国旗问题中more=R+1为什么呢因为荷兰国旗问题是求小于给定值num放在左边，等于放在中间，大于num放在右边，但是快排中或者k小的值的问题中并没有给定这个基准值val，因此需要在数组中指定，一般指定为数组末尾的值，而这个值要固定不变，因此只能交换R前一个位置的值)*

*\* 3.初始化index=L，从L开始遍历数组，知道L<more则遍历结束，因为最终数组会被分为[L,less]小于部分,[less+1,index]等于部分，[index+1,more]大于部分，这三部分。*

*\* 4.arr[index]<val,则index的值和小于等于区域（less）的下一个位置进行交换（即小于等于区域扩了一个位置），然后index+1*

*\* 5.arr[index]>val,则index的值和more的前一个位置进行交换，index的值继续不变*

*\* 6.arr[index]==val，则index++*

*\*/*

**public** **int** **partition(int** L**,int** R**,** **int[]** arr**,** **int** k**){**

**int** less **=** L**-**1**;**

**int** more **=** R**;***//这个地方也是容易出错的地方*

**int** val **=** arr**[**R**];**

**int** index **=** L**;**

**while(**index**<**more**){**

**if(**arr**[**index**]<**val**){***//当前值比index处的值要小*

swap**(**arr**,++**less**,**index**++);**

**}else** **if(**arr**[**index**]>**val**){**

swap**(**arr**,**index**,--**more**);**

**}else{***//当前值和index处的值相等，index++*

index**++;**

**}**

**}**

*//最后交换基准中心位置的值 arr[R]和index位置的值，因为需要将数组 变为左边比arr[R]小右边比arr[R]大*

swap**(**arr**,**index**,**R**);**

System**.**out**.**println**(**"index: "**+**index**);**

**return** index**;**

**}**

**public** **void** **swap(int[]** arr**,int** i**,int** j**){**

**int** tmp **=** arr**[**i**];**

arr**[**i**]** **=** arr**[**j**];**

arr**[**j**]** **=** tmp**;**

**}**

**}**

快排

**package** basic\_class\_01**;**

**import** java.util.Arrays**;**

**public** **class** **Code\_04\_QuickSort** **{**

**public** **static** **void** **quickSort(int[]** arr**)** **{**

**if** **(**arr **==** **null** **||** arr**.**length **<** 2**)** **{**

**return;**

**}**

quickSort**(**arr**,** 0**,** arr**.**length **-** 1**);**

**}**

**public** **static** **void** **quickSort(int[]** arr**,** **int** l**,** **int** r**)** **{**

**if** **(**l **<** r**)** **{**

swap**(**arr**,** l **+** **(int)** **(**Math**.**random**()** **\*** **(**r **-** l **+** 1**)),** r**);**

**int[]** p **=** partition**(**arr**,** l**,** r**);**

quickSort**(**arr**,** l**,** p**[**0**]** **-** 1**);**

quickSort**(**arr**,** p**[**1**]** **+** 1**,** r**);**

**}**

**}**

**public** **static** **int[]** **partition(int[]** arr**,** **int** l**,** **int** r**)** **{**

**int** less **=** l **-** 1**;**

**int** more **=** r**;**

**while** **(**l **<** more**)** **{**

**if** **(**arr**[**l**]** **<** arr**[**r**])** **{**

swap**(**arr**,** **++**less**,** l**++);**

**}** **else** **if** **(**arr**[**l**]** **>** arr**[**r**])** **{**

swap**(**arr**,** **--**more**,** l**);**

**}** **else** **{**

l**++;**

**}**

**}**

swap**(**arr**,** more**,** r**);**

**return** **new** **int[]** **{** less **+** 1**,** more **};**

**}**

**public** **static** **void** **swap(int[]** arr**,** **int** i**,** **int** j**)** **{**

**int** tmp **=** arr**[**i**];**

arr**[**i**]** **=** arr**[**j**];**

arr**[**j**]** **=** tmp**;**

**}**

*// for test*

**public** **static** **void** **comparator(int[]** arr**)** **{**

Arrays**.**sort**(**arr**);**

**}**

*// for test*

**public** **static** **int[]** **generateRandomArray(int** maxSize**,** **int** maxValue**)** **{**

**int[]** arr **=** **new** **int[(int)** **((**maxSize **+** 1**)** **\*** Math**.**random**())];**

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

arr**[**i**]** **=** **(int)** **((**maxValue **+** 1**)** **\*** Math**.**random**())** **-** **(int)** **(**maxValue **\*** Math**.**random**());**

**}**

**return** arr**;**

**}**

*// for test*

**public** **static** **int[]** **copyArray(int[]** arr**)** **{**

**if** **(**arr **==** **null)** **{**

**return** **null;**

**}**

**int[]** res **=** **new** **int[**arr**.**length**];**

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

res**[**i**]** **=** arr**[**i**];**

**}**

**return** res**;**

**}**

*// for test*

**public** **static** **boolean** **isEqual(int[]** arr1**,** **int[]** arr2**)** **{**

**if** **((**arr1 **==** **null** **&&** arr2 **!=** **null)** **||** **(**arr1 **!=** **null** **&&** arr2 **==** **null))** **{**

**return** **false;**

**}**

**if** **(**arr1 **==** **null** **&&** arr2 **==** **null)** **{**

**return** **true;**

**}**

**if** **(**arr1**.**length **!=** arr2**.**length**)** **{**

**return** **false;**

**}**

**for** **(int** i **=** 0**;** i **<** arr1**.**length**;** i**++)** **{**

**if** **(**arr1**[**i**]** **!=** arr2**[**i**])** **{**

**return** **false;**

**}**

**}**

**return** **true;**

**}**

*// for test*

**public** **static** **void** **printArray(int[]** arr**)** **{**

**if** **(**arr **==** **null)** **{**

**return;**

**}**

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

System**.**out**.**print**(**arr**[**i**]** **+** " "**);**

**}**

System**.**out**.**println**();**

**}**

*// for test*

**public** **static** **void** **main(**String**[]** args**)** **{**

**int** testTime **=** 500000**;**

**int** maxSize **=** 100**;**

**int** maxValue **=** 100**;**

**boolean** succeed **=** **true;**

**for** **(int** i **=** 0**;** i **<** testTime**;** i**++)** **{**

**int[]** arr1 **=** generateRandomArray**(**maxSize**,** maxValue**);**

**int[]** arr2 **=** copyArray**(**arr1**);**

quickSort**(**arr1**);**

comparator**(**arr2**);**

**if** **(!**isEqual**(**arr1**,** arr2**))** **{**

succeed **=** **false;**

printArray**(**arr1**);**

printArray**(**arr2**);**

**break;**

**}**

**}**

System**.**out**.**println**(**succeed **?** "Nice!" **:** "Fucking fucked!"**);**

**int[]** arr **=** generateRandomArray**(**maxSize**,** maxValue**);**

printArray**(**arr**);**

quickSort**(**arr**);**

printArray**(**arr**);**

**}**

**}**

荷兰国旗问题

**package** basic\_class\_01**;**

**public** **class** **Code\_08\_NetherlandsFlag** **{**

**public** **static** **int[]** **partition(int[]** arr**,** **int** l**,** **int** r**,** **int** p**)** **{**

**int** less **=** l **-** 1**;**

**int** more **=** r **+** 1**;**

**while** **(**l **<** more**)** **{**

**if** **(**arr**[**l**]** **<** p**)** **{**

swap**(**arr**,** **++**less**,** l**++);**

**}** **else** **if** **(**arr**[**l**]** **>** p**)** **{**

swap**(**arr**,** **--**more**,** l**);**

**}** **else** **{**

l**++;**

**}**

**}**

**return** **new** **int[]** **{** less **+** 1**,** more **-** 1 **};**

**}**

*// for test*

**public** **static** **void** **swap(int[]** arr**,** **int** i**,** **int** j**)** **{**

**int** tmp **=** arr**[**i**];**

arr**[**i**]** **=** arr**[**j**];**

arr**[**j**]** **=** tmp**;**

**}**

*// for test*

**public** **static** **int[]** **generateArray()** **{**

**int[]** arr **=** **new** **int[**10**];**

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

arr**[**i**]** **=** **(int)** **(**Math**.**random**()** **\*** 3**);**

**}**

**return** arr**;**

**}**

*// for test*

**public** **static** **void** **printArray(int[]** arr**)** **{**

**if** **(**arr **==** **null)** **{**

**return;**

**}**

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

System**.**out**.**print**(**arr**[**i**]** **+** " "**);**

**}**

System**.**out**.**println**();**

**}**

**public** **static** **void** **main(**String**[]** args**)** **{**

**int[]** test **=** generateArray**();**

printArray**(**test**);**

**int[]** res **=** partition**(**test**,** 0**,** test**.**length **-** 1**,** 1**);**

printArray**(**test**);**

System**.**out**.**println**(**res**[**0**]);**

System**.**out**.**println**(**res**[**1**]);**

**}**

**}**