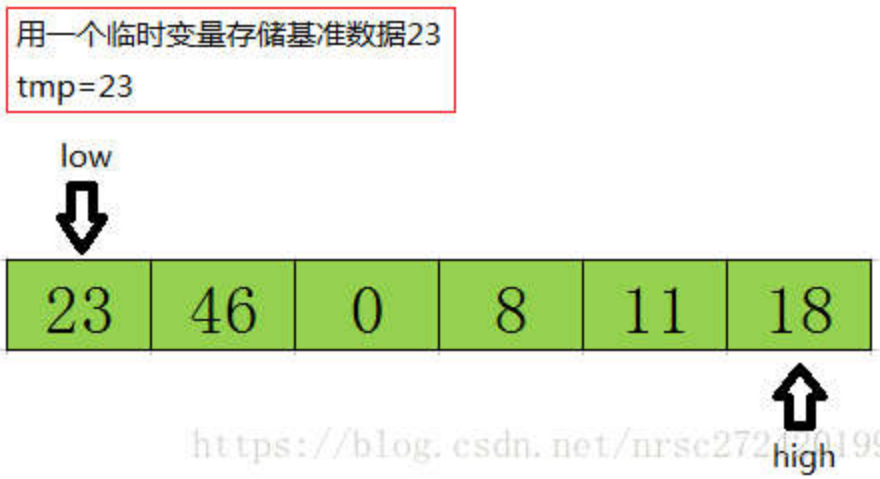
原 快速排序---(面试碰到过好几次)

2018年09月10日 12:20:21 nrsc 阅读数 65828 更多

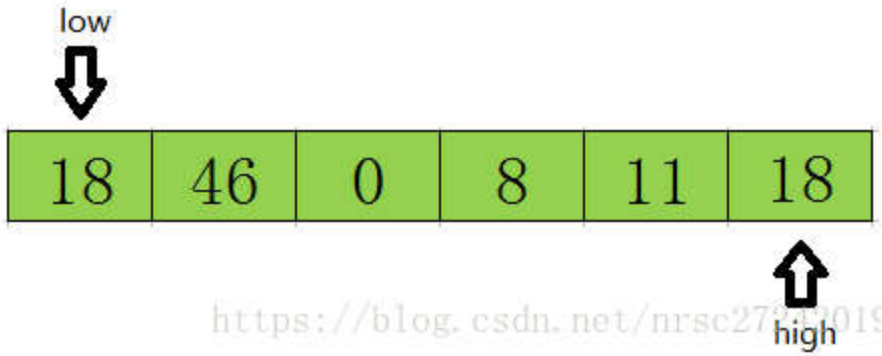
原理:

快速排序,说白了就是给基准数据找其正确索引位置的过程.

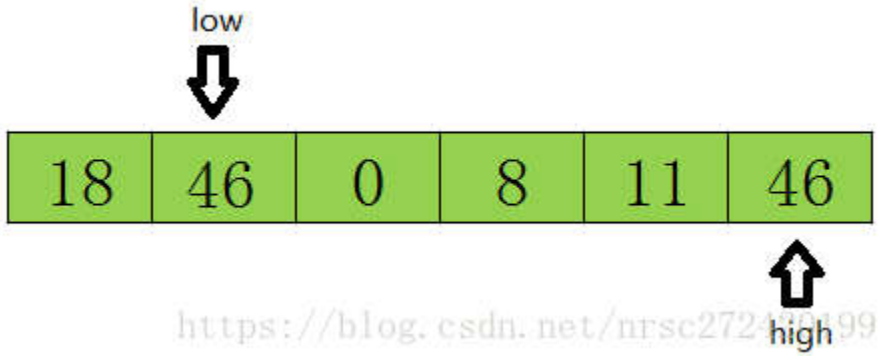
如下图所示,假设最开始的基准数据为数组第一个元素23,则首先用一个临时变量去存储基准数据,即tmp=23;然后分别[illegible]组的两个志:low指向起始位置，high指向末尾.



首先从后半部分开始，**如果扫描到的值大于基准数据就让high减1,如果发现有元素比该基准数据的值小(如上图中18<=tmp)，就**
位置,结果如下:



然后开始从前往后扫描,**如果扫描到的值小于基准数据就让low加1,如果发现有元素大于基准数据的值(如上图46=>tmp)，将low**
指针移动并且数据交换后的结果如下:



然后再开始从后向前扫描,原理同上,发现上图11<=tmp,则将low位置的值赋值给high位置的值,结果如下:

然后再开始从前往后遍历,直到low=high结束循环,此时low或high的下标就是基准数据23在该数组中的正确索引位置.如下图所示.

这样一遍走下来,可以很清楚的知道,其实快速排序的本质就是把基准数大的都放在基准数的左边,把比基准数小的放在基准数的右边,直到数组中的正确位置.

以后采用递归的方式分别对前半部分和后半部分排序，当前半部分和后半部分均有序时该数组就自然有序了。

一些小结论

从上面的过程中可以看到:

①先从队尾开始向前扫描且当low < high时,如果a[high] > tmp,则high–,但如果a[high] < tmp,则将high的值赋值给low,即arr[lo

扫描的方式,即需要从队首开始向队尾进行扫描了

②同理,当从队首开始向队尾进行扫描时,如果a[low] < tmp,则low++,但如果a[low] > tmp了,则就需要将low位置的值赋值给high同时将数组扫描方式换为由队尾向队首进行扫描.

③不断重复①和②,知道low>=high时(其实是low=high),low或high的位置就是该基准数据在数组中的正确索引位置

按照上诉理论我写的代码如下:

```
1  package com.nrsc.sort;
2
3  public class QuickSort {
4      public static void main(String[] args) {
5          int[] arr = { 49, 38, 65, 97, 23, 22, 76, 1, 5, 8, 2, 0, -1, 22 };
6          quickSort(arr, 0, arr.length - 1);
7          System.out.println("排序后:");
8          for (int i : arr) {
9              System.out.println(i);
10         }
11     }
12
13     private static void quickSort(int[] arr, int low, int high) {
14
15         if (low < high) {
16             // 找寻基准数据的正确索引
17             int index = getIndex(arr, low, high);
18
19             // 进行迭代对index之前和之后的数组进行相同的操作使整个数组变成有序
20             quickSort(arr, 0, index - 1);
21             quickSort(arr, index + 1, high);
22         }
23
24     }
```

```
25
26      private static int getIndex(int[] arr, int low, int high) {
27          // 基准数据
28          int tmp = arr[low];
29          while (low < high) {
30              // 当队尾的元素大于等于基准数据时,向前挪动high指针
31              while (low < high && arr[high] >= tmp) {
32                  high--;
33              }
34              // 如果队尾元素小于tmp了,需要将其赋值给low
35              arr[low] = arr[high];
36              // 当队首元素小于等于tmp时,向前挪动low指针
37              while (low < high && arr[low] <= tmp) {
38                  low++;
39              }
40              // 当队首元素大于tmp时,需要将其赋值给high
41              arr[high] = arr[low];
42
43          }
44          // 跳出循环时low和high相等,此时的low或high就是tmp的正确索引位置
45          // 由原理部分可以很清楚的知道low位置的值并不是tmp,所以需要将tmp赋值给arr[low]
46          arr[low] = tmp;
47          return low; // 返回tmp的正确位置
48      }
49  }
50
```


快速排序(三种算法实现和非递归实现) 阅读数 11万+

快速排序(QuickSort)是对冒泡排序的一种改进，基本思想是选取一个记录作为枢轴，经过一趟排序，将... 博文 来自： Maple的博客