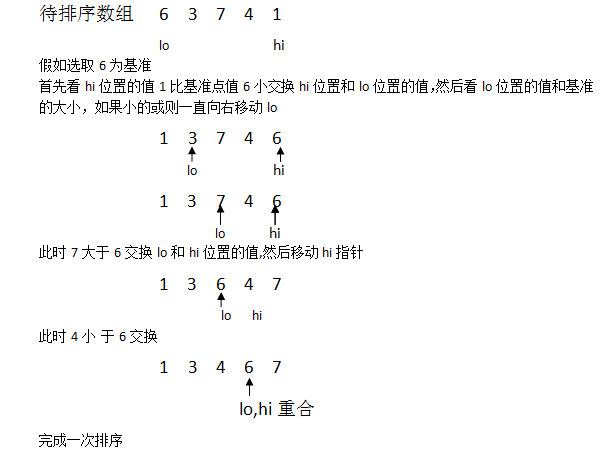
算法思想：基于分治的思想，是冒泡排序的改进型。首先在数组中选择一个基准点（该基准算法思想：基于分治的思想，是冒泡排序的改进型。首先在数组中选择一个基准点（该基准点的选取可能影响快速排序的效率，后面讲解选取的方法），然后分别从数组的两端扫描数组，设两个指示标志（lo指向起始位置，hi指向末尾)，首先从后半部分开始，如果发现有元素比该基准点的值小，就交换lo和hi位置的值，然后从前半部分开始扫秒，发现有元素大于基准点的值，就交换lo和hi位置的值，如此往复循环，直到lo>=hi,然后把基准点的值放到hi这个位置。一次排序就完成了。以后采用递归的方式分别对前半部分和后半部分排序，当前半部分和后半部分均有序时该数组就自然有序了。



**快速排序的时间复杂度为O(NlogN).**

**优化：**

**1.三值取中。**

2. **当待排序序列的长度分割到一定大小后，使用插入排序**

**public class** QuickSort {  
 **public static void** main(String[] args) {  
 **int**[] arrays = {10, 5, 36, 78, 5, 8, 7, 9};  
  
 **int** min = 0;  
 **int** max = arrays.**length**-1;  
 System.***out***.print(**"原来的："**);  
 **for**(**int** i =0;i<=max;i++) {  
 System.***out***.print(arrays[i]+**","**);  
 }  
 System.***out***.println();  
 *sort*(arrays,min,max);  
 System.***out***.print(**"排序后："**);  
 **for**(**int** i =0;i<=max;i++) {  
 System.***out***.print(arrays[i]+**","**);  
 }  
  
 }  
  
 **public static void** sort(**int**[] arrays, **int** min, **int** max) {  
 **if** (min >= max) {  
 **return**;  
 }  
 **int** i = *QuickSort*(arrays, min, max);  
 *sort*(arrays, min, i-1);  
 *sort*(arrays, i+1, max);  
 }  
  
 **public static int** QuickSort(**int**[] arrays, **int** min, **int** max) {  
 **int** key = arrays[min];  
 **while** (min < max) {  
 **while** (min < max && key <= arrays[max]) {  
 max--;  
 }  
*// arrays[min] = arrays[max];  
// arrays[max] = key;  
 swap*(arrays,min,max);  
 **while** (min < max && key >= arrays[min]) {  
 min++;  
 }  
*// arrays[max] = arrays[min];  
// arrays[min] = key;  
 swap*(arrays,min,max);  
 }  
 **return** max;  
 }  
  
 **public static void** swap(**int**[] arrays,**int** a, **int** b) {  
 **int** temp = arrays[a];  
 arrays[a] = arrays[b];  
 arrays[b] = temp;  
 }  
}