排序方法

```
/**
 *
 */
package Easy;


/**
 * @author 64991
 *
 */
public class sortMethods {


    /**
     * @param args
     */


    /*
     *1.选择排序：
    首先，找到数组中最小的那个元素，其次，将它和数组的第一个元素交换位置(如果第一个元素就是最小元素那么它就和自己交换)。其次，在剩下的元素中找到最小的元素，
    将它与数组的第二个元素交换位置。如此往复，直到将整个数组排序。这种方法我们称之为选择排序。*/
    public static int[] selectSort(int [] a) {
        int len = a.length;
        for(int i=0; i<len;i++) {
            int min = i;
```

```
            for(int j=i+1; j<len;j++) {
                if(a[min]>a[j]) {
                    min = j;
                }
                int temp = a[i];
                a[i] = a[min];
                a[min] = temp;

            }
        }
        return a;
    }

    /*
     *2.插入排序：
    1、从数组第 2 个元素开始抽取元素。
    2、把它与左边第一个元素比较，如果左边第一个元素比它大，则继续与左边第二个元素比较下去，
直到遇到不比它大的元素，然后插到这个元素的右边。
    3、继续选取第 3，4，….n 个元素,重复步骤 2 ，选择适当的位置插入。
    插入排序是稳定的*/
    public static int[] insertSort(int[] a) {
        if(a == null || a.length == 1) {
            return a;
        }
        for(int i=1;i<a.length;i++) {
```

```
            int k = i;
            while(k>0 && a[k]>a[k-1]) {
                int temp = a[k];
                a[k] = a[k-1];
                a[k-1] = temp;
                k--;
        }
        }
        return a;
    }

    /*
     *3、冒泡排序
1、把第一个元素与第二个元素比较，如果第一个比第二个大，则交换他们的位置。
接着继续比较第二个与第三个元素，如果第二个比第三个大，则交换他们的位置….
我们对每一对相邻元素作同样的工作，从开始第一对到结尾的最后一对，这样一趟比较交换下来之后，
排在最右的元素就会是最大的数。
除去最右的元素，我们对剩余的元素做同样的工作，如此重复下去，直到排序完成。
性质：1、时间复杂度：O(n2)  2、空间复杂度：O(1)  3、稳定排序  4、原地排序*/
    public static int[] bubbleSort(int[] a) {
        if(a == null || a.length == 1) {
            return a;
        }
        for(int i=0;i<a.length;i++) {
```

```
            boolean flag = true;
            for(int j = 0; j < a.length - i - 1; j++) {
                if(a[j+1] < a[j]) {
                    flag = false;
                    int temp = a[j+1];
                    a[j+1] = a[j];
                    a[j] = temp;
                }
            }
            if(flag) break;
        }
        return a;
    }
```

/*4、快速排序

我们从数组中选择一个元素，我们把这个元素称之为中轴元素吧，然后把数组中所有小于中轴元素的元素放在

其左边，所有大于或等于中轴元素的元素放在其右边，显然，此时中轴元素所处的位置的是有序的。

也就是说，我们无需再移动中轴元素的位置。

从中轴元素那里开始把大的数组切割成两个小的数组(两个数组都不包含中轴元素)接着我们通过递归的方式

，让中轴元素左边的数组和右边的数组也重复同样的操作，直到数组的大小为 1，

此时每个元素都处于有序的位置。

性质：1、时间复杂度：O(nlogn)　2、空间复杂度：O(logn)　3、非稳定排序　4、原地排序*/

```
public static int[] quickSort(int[] a, int left, int right) {

    if(left<right) {
        int center = partition2(a, left, right);
        a = quickSort(a, left, center-1);
        a = quickSort(a, center+1,right);
    }
    return a;
}

//这样分组会有比较多的重复比较，因此还需要改进一下
public static int partition(int[] a, int left, int right) {
    int temp;
    int i = left;
    int j;
    int pivot = a[right];
    for(j = left; j < right; j++) {
        if(a[j]<pivot) {
            temp = a[i];
            a[i] = a[j];
            a[j] = temp;
            i++;
        }
    }
    a[right] = a[i];
```

```
        a[i] = pivot;
        return i;
    }

    public static int partition2(int[] a, int left, int right) {
        int i = left+1;
        int j = right;
        int pivot = a[left];
        while(true) {
            while(i <= j && a[i] <= pivot) i++;
            while(i <= j && a[j] >= pivot) j--;
            if(i >= j) break;
            int temp = a[j];
            a[j] = a[i];
            a[i] = temp;

        }
        a[left] = a[j];//或者 A[I]
        a[j] = pivot;
        return j;
    }

    /*
     * 5、归并排序
    将一个大的无序数组有序，我们可以把大的数组分成两个，然后对这两个数组分别进行排序，
```

之后在把这两个数组合并成一个有序的数组。由于两个小的数组都是有序的，所以在合并的时候是很快的。

通过递归的方式将大的数组一直分割，直到数组的大小为 1，此时只有一个元素，

那么该数组就是有序的了，之后再把两个数组大小为 1 的合并成一个大小为 2 的，

再把两个大小为 2 的合并成 4 的 ..... 直到全部小的数组合并起来。

性质：1、时间复杂度：O(nlogn)　2、空间复杂度：O(n)　3、稳定排序　4、非原地排序*/

```
//递归版本
public static int[] mergeSort(int[] a, int left, int right) {
    //如果 left == right，表示数组只有一个元素，则不用递归排序
    if(left < right) {
        int mid = (right + left)/2;
        a = mergeSort(a, left, mid);
        a = mergeSort(a, mid+1, right);
        merge(a, left, mid, right);
    }
    return a;
}

//非递归
public static int[] mergeSort2(int[] a) {
    int n = a.length;
     // 子数组的大小分别为 1，2，4，8...
     // 刚开始合并的数组大小是 1，接着是 2，接着 4....
    for(int i = 1; i < n; i+=i) {
```

```
            int left = 0;
            int mid = left + i - 1;
            int right = mid + i;
            while(right < n) {
                merge(a, left, mid, right);
                left = right + 1;
                mid = left + i -1;
                right = mid + i;
            }
            if(left < n && mid < n) {
                merge(a, left, mid, n-1);
            }
        }

        return a;
}

public static void merge(int[] a, int left, int mid,int right) {
        int b[] = new int[right-left+1];
        int i = left;
        int j = mid + 1;
        int k = 0;
        while(i <= mid && j <= right) {
            if(a[i] < a[j]) {
                b[k++] = a[i++];
```

```
            }else b[k++] = a[j++];
        }
          while(i <= mid) b[k++] = a[i++];
          while(j <= right) b[k++] = a[j++];
        //临时数组复制到原数组
        for(i = 0; i < k; i++) {
            a[left++] = b[i];
        }
    }

    /*
     * 6、堆排序
堆的特点就是堆顶的元素是一个最值，大顶堆的堆顶是最大值，小顶堆则是最小值。
堆排序就是把堆顶的元素与最后一个元素交换，交换之后破坏了堆的特性，
我们再把堆中剩余的元素再次构成一个大顶堆，然后再把堆顶元素与最后第二个元素交换....如此往复下去，
等到剩余的元素只有一个的时候，此时的数组就是有序的了。*/

    public static int[] heapSort(int[] a) {
        int length = a.length;
        //构建二叉堆
        for(int i = (length-2)/2;i >= 0;i--) {
            a = heapify(a, i, length);
        }

        //开启堆排序
```

```
        for(int i = length - 1; i >= 0; i--) {
            int temp = a[i];
            a[i] = a[0];
            a[0] = temp;
            a = heapify(a, 0, i);
        }
        return a;
    }

    public static int[] heapify(int a[], int parent, int length) {
        int temp = a[parent];
        int lchild = 2 * parent + 1;
        while(lchild < length) {
            if(lchild + 1 < length && a[lchild] < a[lchild+1]) {
                lchild++;
            }
            if(temp > a[lchild]) break;

            a[parent] = a[lchild];
            parent = lchild;
            lchild = 2 * parent + 1;

        }
        a[parent] = temp;
        return a;
```

```
    }

    /*
     * 8、计数排序
    计数排序是一种适合于最大值和最小值的差值不是不是很大的排序。

    基本思想：就是把数组元素作为数组的下标，然后用一个临时数组统计该元素出现的次数，
    例如 temp[i] = m, 表示元素 i 一共出现了 m 次。最后再把临时数组统计的数据从小到大汇总
    起来，此时汇总起来是数据是有序的。*/
    public static int[] countSort(int[] a) {
        if(a == null || a.length<2) {
            return a;
        }
        int length = a.length;
        int min = a[0];
        int max = a[0];

        for(int i = 0; i < length; i++) {
            if(a[i] > max) max = a[i];
            if(a[i] < min) min = a[i];
        }
        int d = max - min + 1;
        //2.创建统计数组并统计对应元素个数
        int[] temp = new int[d];
```

```
        for(int i = 0; i < length; i++) {
            temp[a[i]-min]++;
        }

        //3.统计数组做变形，后面的元素等于前面的元素之和
        int k = 0;
        for(int i = 0; i < d; i++) {
            for(int j = temp[i]; j > 0; j--) {
                a[k++] = i + min;
            }
        }
        return a;
    }

    public static void main(String[] args) {
        int a[] = {6,7,8,5,4,3,2,1};
        a=countSort(a);
        for(int x:a) {
        System.out.print(x);
        }
    }

}
```