数组的排序方法

1. **冒泡排序、快速排序、希尔排序、选择排序**
2. 基本思想介绍：
3. 冒泡排序：

基本思想：在要排序的一组数中，对当前还未排好序的范围内的全部数，自上而下对相邻的两个数依次进行比较和调整，让较大的数往下沉，较小的往上冒。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。**

1. 快速排序

基本思想：**选择一个基准元素**,通常选择第一个元素或者最后一个元素,通过一趟扫描，将待排序列分成两部分,一部分比基准元素小,一部分大于等于基准元素,此时基准元素在其排好序后的正确位置,然后**再用同样的方法递归地排序划分的两部分**。

1. 希尔排序
2. 选择排序
3. **直接插入排序**：

基本思想：在要排序的一组数中，假设前面(n-1)[n>=2] 个数已经是排好顺序的，现在要把第n个数插到前面的有序数中，使得这n个数也是排好顺序的。如此反复循环，直到全部排好顺序

package com.firewolf.sort;

public class MySort {

    /\*\*

     \* @param args

     \*/

    public static void main(String[] args) {

        int array[] = {45,32,54,12,43,65,11,3,43,6,33,90,44,1,178};

        MySort mySort = new MySort();

        mySort.insertSort(array);

        System.out.print("插入排序结果 ：  ");

        mySort.printArray(array);

        System.out.println();

        mySort.bubbleSort(array);

        System.out.print("冒泡排序结果 ：  ");

        mySort.printArray(array);

        mySort.qsort(array);

        System.out.println();

        System.out.print("快速排序结果 ：  ");

        mySort.printArray(array);

        mySort.shellSort(array);

        System.out.println();

        System.out.print("希尔排序结果 ：  ");

        mySort.printArray(array);

        mySort.selectSort(array);

        System.out.println();

        System.out.print("选择排序结果 ：  ");

        mySort.printArray(array);

    }

    /\*\*

      \* 直接插入排序

      \* 基本思想：在要排序的一组数中，假设前面(n-1)[n>=2] 个数已经是排好顺序的，现在要把第n个数插到前面的有序数中，使得这n个数也是排好顺序的。如此反复循环，直到全部排好顺序

     \*/

    public void insertSort(int[] array){

        int temp=0;

        for(int i=1;i<array.length;i++){

           int j=i-1;

           temp=array[i];

           for(;j>=0&&temp<array[j];j--){

               array[j+1]=array[j];                       //将大于temp的值整体后移一个单位

           }

           array[j+1]=temp;

        }

    }

    /\*\*

      \* 冒泡排序

      \* 基本思想：在要排序的一组数中，对当前还未排好序的范围内的全部数，自上而下对相邻的两个数依次进行比较和调整，让较大的数往下沉，较小的往上冒。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。

     \*/

    public void bubbleSort(int[] array) {

        int temp;

        for(int i=0;i<array.length;i++){//趟数

          for(int j=0;j<array.length-i-1;j++){//比较次数

            if(array[j]>array[j+1]){

              temp=array[j];

              array[j]=array[j+1];

              array[j+1]=temp;

            }

          }

        }

    }

    /\*\*

      \* 快速排序

      \* 基本思想：选择一个基准元素,通常选择第一个元素或者最后一个元素,通过一趟扫描，将待排序列分成两部分,一部分比基准元素小,一部分大于等于基准元素,此时基准元素在其排好序后的正确位置,然后再用同样的方法递归地排序划分的两部分。

     \* @param array

     \*/

    public void qsort(int array[]){

        if(array.length>1){

            \_qsort(array,0,array.length-1);

        }

    }

    /\*\*

     \* 一趟快速排序

     \* @param array

     \*/

    private void \_qsort(int[] array,int low,int high){

        if(low < high){

            int middle = getMiddle(array, low, high);

            \_qsort(array,low,middle-1);

            \_qsort(array, middle+1, high);

        }

    }

    /\*\*

     \* 得到中间值

     \*/

    private int getMiddle(int[] array,int low,int high){

        int tmp = array[low];

        while(low < high){

            while(low < high && **array[high] >= tmp**)

                high--;

            array[low] = array[high];

            while(low<high && **array[low]<=tmp**)

                low++;

            array[high] = array[low];

        }

        array[low] = tmp;

        return low;

    }

    /\*\*

**\* 简单选择排序**

**\* 基本思想：在要排序的一组数中，选出最小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的与第二个位置的数交换，如此循环到倒数第二个数和最后一个数比较为止。**

     \* @param array

     \*/

    public void selectSort(int[] array){

        int position=0;

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

            int j=i+1;

            position=i;

            int temp=array[i];

            for(;j<array.length;j++){

            if(array[j]<temp){

                temp=array[j];

                position=j;

            }

            }

            array[position]=array[i];

            array[i]=temp;

        }

    }

    /\*\*

     \* 希尔排序（最小增量排序）

     \* 基本思想：算法先将要排序的一组数按某个增量d（n/2,n为要排序数的个数）分成若干组，每组中记录的下标相差d.对每组中全部元素进行直接插入排序，然后再用一个较小的增量（d/2）对它进行分组，在每组中再进行直接插入排序。当增量减到1时，进行直接插入排序后，排序完成。

     \* @param array

     \*/

    public  void shellSort(int[] array){

        double d1=array.length;

        int temp=0;

        while(true){

            d1= Math.ceil(d1/2);

            int d=(int) d1;

            for(int x=0;x<d;x++){

                for(int i=x+d;i<array.length;i+=d){

                    int j=i-d;

                    temp=array[i];

                    for(;j>=0&&temp<array[j];j-=d){

                        array[j+d]=array[j];

                    }

                    array[j+d]=temp;

                }

            }

            if(d==1)

                break;

        }

    }

    /\*\*

     \* 打印数组中的所有元素

     \*/

    public void printArray(int[] array){

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

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

        }

    }

}