**内部排序**

**一．直接插入排序**

1.概念

直接插入排序（Straight Insertion Sort）是一种最简单的排序方法，其基本操作是将一条记录插入到已排好的有序表中，从而得到一个新的、记录数量增1的有序表。

1. 直接插入排序(straight insertion sort)的做法是：  
   每次从无序表中取出第一个元素，把它插入到有序表的合适位置，使有序表仍然有序。  
   第一趟比较前两个数，然后把第二个数按大小插入到有序表中； 第二趟把第三个数据与前两个数从后向前扫描，把第三个数按大小插入到有序表中；依次进行下去，进行了(n-1)趟扫描以后就完成了整个排序过程。
2. 直接插入排序是由两层嵌套循环组成的。外层循环标识并决定待比较的数值。内层循环为待比较数值确定其最终位置。直接插入排序是将待比较的数值与它的前一个数值进行比较，所以外层循环是从第二个数值开始的。当前一数值比待比较数值大的情况下继续循环比较，直到找到比待比较数值小的并将待比较数值置入其后一位置，结束该次循环。
3. 最好情况：即待排序列已按关键码有序，每趟操作只需一次比较，二次移动。总比较次数为n-1次，总移动次数为2（n-1）次。

最坏情况：即第j套操作，插入记录需要同前面的j个记录进行，j次关键码比较，移动记录的次数为j+2次。总比较次数为n(n-1) /2，总移动次数为n(n-1) /2+2n.

平均情况：既第j 趟操作，插入记录大约通前面的j/2个记录进行关键码比较，移动记录的次数为j/2+2次。总比较次数为n(n-1)/4+2n≈n2/4.

由此可知，直接插入排序的时间复杂度为O（n2）.

**二．折半插入排序**

1.概念

折半插入排序（Binary Insertion Sort）是对插入排序算法的一种改进，所谓排序算法过程，就是不断的依次将元素插入前面已排好序的序列中。

1. 排序思想：

有一组数据待排序，排序区间为Array[0]~Array[n-1]。将数据分为有序数据和无序数据，第一次排序时默认Array[0]为有序数据，Array[1]~Array[n-1]为无序数据。有序数据分区的第一个元素位置为low，最后一个元素的位置为high。

1. 遍历无序区间的所有元素，每次取无序区间的第一个元素Array[i]，因为0~i-1是有序排列的，所以用中点m将其平分为两部分，然后将待排序数据同中间位置为m的数据进行比较，若待排序数据较大，则low~m-1分区的数据都比待排序数据小，反之，若待排序数据较小，则m+1~high分区的数据都比 待排序数据大，此时将low或high重新定义为新的合适分区的边界，对新的小分区重复上面操作。直到low和high 的前后顺序改变，此时high+1所处位置为待排序数据的合适位置。
2. 优缺点

优点 : 稳定，相对于直接插入排序元素减少了比较次数；

缺点 : 相对于直接插入排序元素的移动次数不变。

1. 时间复杂度

可以看出，折半插入排序减少了比较元素的次数，约为O(nlogn)，比较的次数取决于表的元素个数n。因此，折半插入排序的时间复杂度仍然为O(n²)，但它的效果还是比直接插入排序要好。

6.空间复杂度：排序只需要一个位置来暂存元素，因此空间复杂度为O（1）。

**三．冒号排序**

**1.概念**

冒泡排序是排序算法的其中一种，该排序的逻辑理解起来较为容易，理解上可以有两种方式，一种中正向的思维，一种是逆向的思维，所谓的正向思维就是从前往后，从左往右，从上到下。那么逆向思维呢就正好与之相反。

下面来说一正向思维下的冒泡排序：

例如给你一组数据：{1, 34, 56, 8, -32, 7, -9, 0, 235 }在正向思维下的排序方式就是从左到右的进行排序，其排序的是按照第一个数和第二个数比较大小，如果第一个数比第二个数大的话，第二个数就和第一个数交换位置，第二个在和第三个比较大小，如果第二个数比第三个数小那么就位置不变，反之就交换位置，接着往后比较，依次进行下去。总结的来说就是进行一次完整的排序之后就会出现一个现象就是排在最后的数最大。

例如： 1, 34, 56, 8, -32, 7, -9, 0, 235

第一次：1, 34, 8,-32 ,7 , -9, 0, 56, 235 235最大的排在了最后；

第二次：1,8，-32,7,-9,0,34,56,235 56除235之外的最大的排在了最后；

...

第八次：-32,-9,0,1,7,8,34,56,235 ...

代码：/\* 正向思维下的冒泡排序 \*/

public class MaoPaoSort {

public static void main(String[] args) {

System.out.print("[");

int nums[] = { 1, 34, 56, 8, -32, 7, -9, 0, 235 };

for (int i = 0; i < nums.length; i++) {// 当前要确定的是哪个位置的数

for (int j = 1; j < nums.length - 1 - i; j++) {

if (nums[j - 1] > nums[j]) {

int tmp = nums[j - 1];

nums[j - 1] = nums[j];

nums[j] = tmp;

}

}

}

for (int x : nums) { System.out.print(x + ","); }

System.out.print("]")； }

}

逆向思维与正向思维下的相反：

/\*逆向思维下的冒泡排序\*/

public class MaoPaoSort {

   public static void main(String[] args) {

   int [] arr={1, 34, 56, 8, -32, 7, -9, 0, 235};

   for(int i=0;i<arr.length;i++){

   for(int j=arr.length-1;j>=i+1;j--){

   if(arr[j]<arr[j-1])

{ int t=arr[j];

       arr[j]=arr[j-1];

       arr[j-1]=t;}}

   }

    for(int x:arr){System.out.println(x);}

    }

}