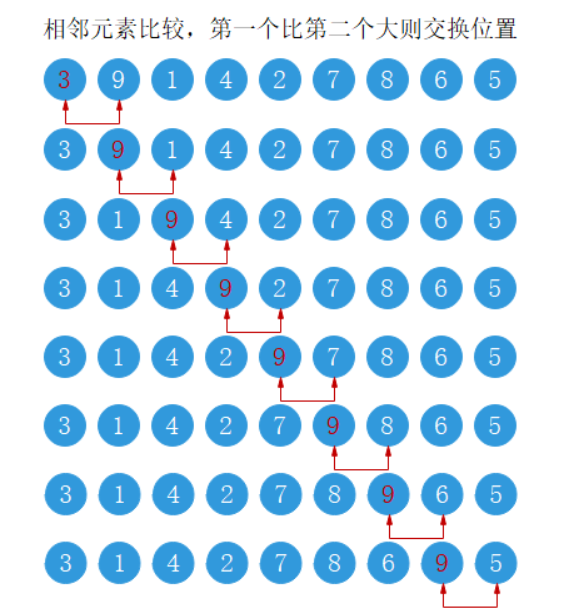
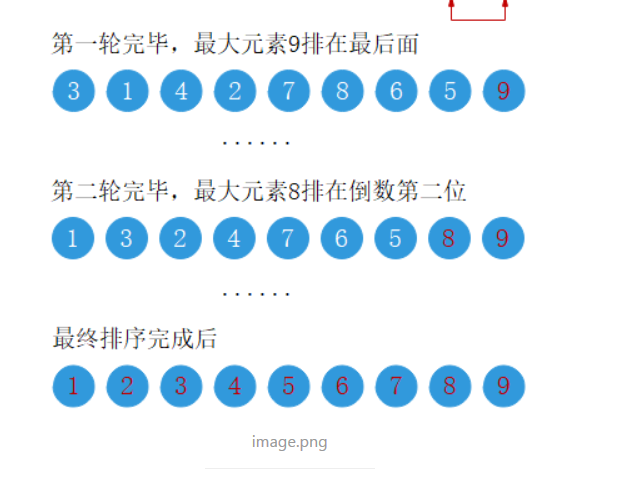
原理：如同冒泡排序的名字一样，像汽水中的气泡会浮到顶端一样，此排序的每一次过程都会将最大元素浮到顶端。(从大到小)

实现过程：

1. 比较相邻元素，如果第一个比第二个大就交换位置。
2. 每一轮对每一对相邻的元素做同样的操作，直到最大值到达顶端
3. N个数两两相互比较每一轮并把最大值放在顶端因此只需要比较n-1轮
4. 由于每一轮比较的时候都会把最大的放在最顶端，因此下一轮比较只要到达其前一个就停止。因此每一轮比较个数n-i(i是第几轮)





//伪算法

Public function($data){

$len=count($data);

For($i=1;$i<len;$i++){//进行n-1轮冒泡

For($j=0;$j<len-$i;$j++)//剩余乱序的数

{

If($data[$j]>$data[$j+1])

{

$temp=$data[$j];

$data[$j]=$data[$j+1];

$data[$j+1]=$temp;

}

}

Return $data;

}

冒泡算法最好的情况正序

只需要进行n-1次比对

时间复杂度O(n)

最坏情况反序

执行n-1趟和n-i此比对

平均时间复杂度==O（N\*\*）

时间复杂度，空间复杂度计算过程：<https://blog.csdn.net/jsjwk/article/details/84315770>

果n无限大的时候，T(n) =  time(1+2n)中的常量1就没有意义了，倍数2也意义不大。因此直接简化为T(n) =  O(n) 就可以了。

时间复杂度计算：T(n) = O( f(n) )，其中f(n) 表示每行代码执行次数之和

两层循环，第1次遍历n次(n个元素)，第二次遍历n-1次，... 依次类推

N+(n-1)+(n-2)+…+1=n\*(n+1)/2=O(n\*\*);

空间复杂度： O（1）

空间复杂度是对一个算法在运行过程中临时占用存储空间大小的一个量度，

如果没有new 数组都是O(1)

临时New 长度为n的数组 空间复杂度O(n)