1. **冒泡排序（Bubble Sort）**

它重复地走访过要排序的数列，依次比较两个元素，如果他们的顺序错误就把他们交换过来；走访数列的工作是重复地进行直到没有再需要交换；这个算法的名字由来是因为越小的元素会经由交换慢慢“浮”到数列的顶端

与前面的比较，交换与否

1. 步骤
2. 比较相邻的元素：如果第一个比第二个大，就交换他们两个
3. 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，最后的元素应该会是最大的数
4. 针对所有的元素重复以上的步骤，除了最后一个
5. 持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字 需要比较
6. 代码



1. **快速排序**

快速排序是由东尼·霍尔所发展的一种排序算法。在平均状况下，排序 n 个项目要Ο(n log n)次比较。在最坏状况下则需要Ο(n2)次比较，但这种状况并不常见。事实上，快速排序通常明显比其他Ο(n log n) 算法更快，因为它的内部循环（inner loop）可以在大部分的架构上很有效率地被实现出来，且在大部分真实世界的数据，可以决定设计的选择，减少所需时间的二次方项之可能性

重点是基准值，递归

1. 步骤
2. 从数列中挑出一个元素，称为 “基准”（pivot）
3. 重新排序数列：所有元素比基准值小的摆放在基准前面，所有元素比基准值大的摆在基准的后面（相同的数可以到任一边）；在这个分区退出之后，该基准就处于数列的中间位置。这个称为分区（partition）操作
4. 递归地（recursive）把小于基准值元素的子数列和大于基准值元素的子数列排序
5. 代码



1. **选择排序(Selection sort)**

实现思路：双重循环完成，外层控制轮数，当前的最小值；内层控制的比较次数

重点在最小值

1. 步骤
2. 首先在未排序序列中找到最小元素，存放到排序序列的起始位置
3. 再从剩余未排序元素中继续寻找最小元素，然后放到排序序列末尾
4. 以此类推，直到所有元素均排序完毕
5. 代码



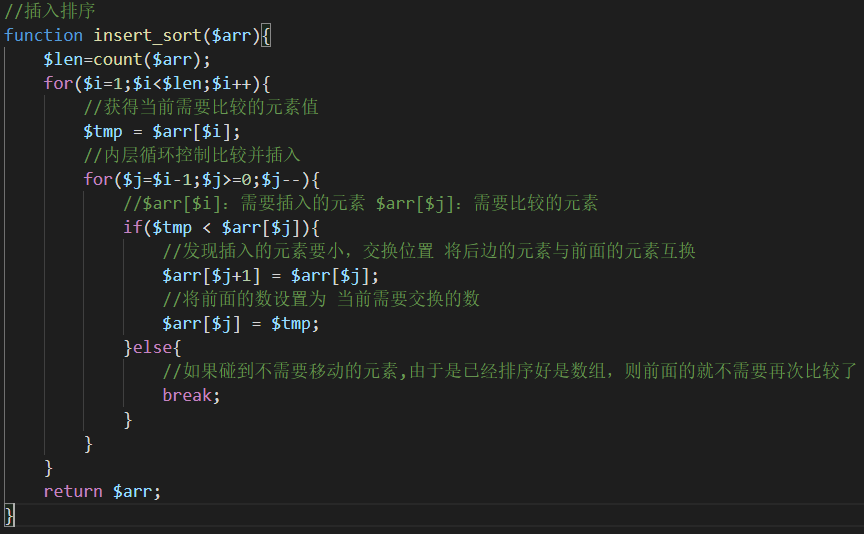
1. **插入排序（Insertion Sort）**

工作原理是通过构建有序序列，对于未排序数据，在已排序序列中从后向前扫描，找到相应位置并插入。

插入排序在实现上，通常采用in-place排序（即只需用到O(1)的额外空间的排序），因而在从后向前扫描过程中，需要反复把已排序元素逐步向后挪位，为最新元素提供插入空间

遍历元素与前面的元素比较，根据结果执行交换与否，直到遇到否的情况则终止内部遍历

1. 步骤
2. 从第一个元素开始，该元素可以认为已经被排序
3. 取出下一个元素，在已经排序的元素序列中从后向前扫描
4. 如果该元素（已排序）大于新元素，将该元素移到下一位置
5. 重复步骤3，直到找到已排序的元素小于或者等于新元素的位置
6. 将新元素插入到该位置中
7. 重复步骤2
8. 代码



1. **其他**