插入排序：

直接插入排序：

插入排序是最简单的排序算法之一，对于有N个元素的序列，插入排序由N-1趟排序组成。它的工作原理是通过构建有序序列，对于未排序的数据，在已经排序序列中从后向前扫描，找到相应位置插入。

直接插入排序算法步骤：

1. 将第一个待排序的序列的第一个元素看做一个有序序列，把第二个元素到最后一个元素当成是未排序序列。
2. 从头到尾一次扫描未排序的序列，将扫描到的每个元素插入有序序列的适当位置（在这里需要注意一个问题，如果在有序序列中有一个和待插入的元素相等，则将待插入的元素查到此元素的后面，这样方式的插入排序是稳定的。如果插入到此元素的前面，那么此种方式的插入排序是不稳定的）

对于上面的步骤可以将插入排序的算法总结为：

在第L趟排序中，将位置L上的元素向左移动到它在前L+1个元素中的正确的位置上，这里的前L个元素是有序的。

如下图：

初始序列： 3 6 8 7 2 4 9 1 5

第一趟排序：

在位置1上的元素6 可将前面的1个元素看做有序序列

开始

3 6 8 7 2 4 9 1 5

结束

3 6 8 7 2 4 9 1 5

在位置2上的元素8 前面的2个元素有序，

第二趟排序：

开始

3 6 8 7 2 4 9 1 5

结束

3 6 8 7 2 4 9 1 5

在位置3上的元素7 前面的3个元素有序，将7 插入到6和8之间

开始

3 6 8 7 2 4 9 1 5

第三趟排序：

结束

3 6 7 8 2 4 9 1 5

第七趟排序：

在位置7上的元素1 前面的7个元素有序，将1 插入到2之前

开始

2 3 4 6 7 8 9 1 5

结束

1 2 3 4 6 7 8 9 5

开始

1 2 3 4 6 7 8 9 5

第八趟排序：

在位置8上的元素5 前面的8个元素有序，将5 插入到4和6之间

结束

1 2 3 4 5 6 7 8 9

插入排序算法分析：

由于嵌套循环的每一个都花费N次迭代，因此插入排序的时间复杂度为O(N^2);

以上讲解的插入排序是插入排序算法中的直接插入排序，在有序序列中查找插入位置的时候是逐个查找，故称直接插入排序

欢迎继续关注我们后续的排序算法

折半插入排序：

插入排序是最简单的排序算法之一，对于有N个元素的序列，插入排序由N-1趟排序组成。它的工作原理是通过构建有序序列，对于未排序的数据，在已经排序序列中从后向前扫描，找到相应位置插入。

折半插入排序算法步骤

1. 将第一个待排序的序列的第一个元素看做一个有序序列，把第二个元素到最后一个元素当成是未排序序列。
2. 从头到尾一次扫描未排序的序列，将扫描到的每个元素插入有序序列的适当位置，在这里和直接插入排序算法不同的是，折半插入排序根据二分查找法在有序序列中查找合适的位置将还未排序的元素插入。其它方面和直接插入排序是相同的（在这里需要注意一个问题，如果在有序序列中有一个和待插入的元素相等，则将待插入的元素查到此元素的后面，这样方式的插入排序是稳定的。如果插入到此元素的前面，那么此种方式的插入排序是不稳定的）

插入排序算法分析：

由于嵌套循环的每一个都花费N次迭代，因此插入排序的时间复杂度为O(N^2);

因为虽然在查找的时候采用了二分查找法，查找的时间复杂度为O（nlogn）但是在查找到合适的位置以后，移动元素同样也需要n次，所以折半插入排序的时间复杂度也是O（N^2）

下面奉上折半插入排序的php代码

<?php

$array=array(3,6,4,8,1,5,9,2,7);

for($i=1;$i<count($array);$i++){

if($array[$i]<$array[$i-1]){

$pos=getpos(0,$i-1,$array[$i],$array);

$ta=$array[$i];

if($array[$pos]>$ta){

$pos=$pos-1;

}

for($j=$i-1;$j>$pos;$j--){

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

}

$array[$j+1]=$ta;

}

}

function getpos($low,$high,$data,$arr){

$mid=ceil(($low+$high)/2)-1;

while($mid>$low){

if($arr[$mid]>=$data){

$high=$mid;

}else{

$low=$mid;

}

$mid=ceil(($low+$high)/2)-1;

}

return $mid;

}

希尔排序

希尔排序是按照该算法的设计者的名字 希尔 命名的。现在希尔排序一直是教科书和参考手册中所重点突出的算法。

希尔排序的产生是他的设计者希尔基于插入排序的性质改进的：

第一点：插入排序算法对于已经有序的数据进行排序操作的时候，效率很高，可以达到线性排序的效率。

第二点：插入排序进行排序的时候，每一趟排序只能移动一个数据。所以说这样的排序方法相对来说又是低效率的。

对于希尔排序，用一句话来说，实质上是一种分组插入方法，是递减增量排序算法。

希尔排序的思想是：先将整个待排序的记录序列分割成为若干子序列分别进行直接插入排序，待整个序列中的记录基本上有序的时候，再对全体记录进行依次直接插入排序，因为直接插入排序在元素基本有序的情况下（根据上述第一点，接近最好的情况），效率是很高的。

希尔排序算法步骤：

1. 选择一个增量序列 a1,a2,…,ak, 其中 ai>aj , ak=1;
2. 按增量序列个数k，对序列进行k趟排序；
3. 每趟排序，根据对应的增量ai，将待排序列分割成若干长度为m的子序列，分别对各子表进行直接插入排序。仅增量因子为1时，整个序列作为一个表来处理表长度即为整个序列的长度。

形象化为下图：

3 7 2 9 0 6 1 5 4 6 8

5

第一趟排序前

3 1 2 4 0 6 7 5 9 6 8

第一趟排序后

0 1 2 4 3 5 7 6 8 6 9

第二趟排序后

第二趟排序前

3 1 2 4 0 6 7 5 9 6 8

2

第三趟排序后

0 1 2 3 4 5 6 6 7 8 9

第三趟排序前

0 1 2 4 3 5 7 6 8 6 9

1

由以上的排序图也可以看出 希尔排序是稳定排序

下面奉上希尔排序的php代码

$arr=array(3,7,2,9,0,6,1,5,4,6,8);

$pos=false;

for($dl=floor(count($arr)/2);$dl>=1;$dl=floor($dl/2)){

for($i=$dl;$i<count($arr);$i++) {

for($j=$i-$dl;$j>=0;$j-=$dl){

if($arr[$j+$dl]<$arr[$j]){

$temp=$arr[$j];

$arr[$j]=$arr[$j+$dl];

$arr[$j+$dl]=$temp;

}

}

}

}

快速排序

快速排序是由东尼•霍尔所发展的一种排序算法，它是对冒泡排序的一种改进。

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。

快速排序使用分治法（Divide and conquer）策略来把一个串行（list）分为两个子串行（sub-lists）。

它的基本思想是：通过一趟排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一部分的所有数据都要小，然后再按此方法对这两部分数据分别进行快速排序，整个排序过程可以递归进行，以此达到整个数据变成有序序列。

快速排序算法步骤

1 从数列中挑出一个元素，称为 “基准”（pivot），

2 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。

3 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。

递归的最底部情形，是数列的大小是零或一，也就是永远都已经被排序好了。虽然一直递归下去，但是这个算法总会退出，因为在每次的迭代（iteration）中，它至少会把一个元素摆到它最后的位置去。

对于一趟快速排序的算法步骤：

1）设置两个变量low、high，排序开始的时候：low=0，high=N-1；

2）以第一个数组元素作为关键数据，赋值给key，即key=A[0]；

3）从high开始向前搜索，即由后开始向前搜索(high--)，找到第一个小于key的值A[high]，将A[high]和A[low]互换；

4）从low开始向后搜索，即由前开始向后搜索(low++)，找到第一个大于key的A[low]，将A[low]和A[high]互换；

5）重复第3、4步，直到low=high； (3,4步中，没找到符合条件的值，即3中A[high]不小于key,4中A[low]不大于key的时候改变high 、low的值，使得high = high -1，lowi=low+1，直至找到为止。找到符合条件的值，进行交换的时候low， high指针位置不变。另外，low==high这一过程一定正好是low+或high-完成的时候，此时令循环结束）。

值得注意的是，快速排序不是一种稳定的排序算法，也就是说，多个相同的值的相对位置也许会在算法结束时产生变动。

对以下数组进行快速排序，下图示出了排序过程

交换

1 5 2 6 8 3 7 4

Low=0

high=5

3 5 2 6 8 1 7 4

high=7

Low=0

3 5 2 6 8 1 7 4

3 5 2 6 8 1 7 4

Key=3

high=2

Low=1

Low=1

1 2 3 6 8 5 7 4

交换

1 3 2 6 8 3 7 4

1 3 2 6 8 5 7 4

交换

1 5 2 6 8 3 7 4

high=5

第一趟排序

1 2 3 4 6 5 7 8

1 2 3 4 6 5 7 8

交换

Low=4

high=7

1 2 3 4 8 5 7 6

1 2 3 4 8 5 7 6

交换

high=7

Low=3

1 2 3 6 8 5 7 4

因partion左边的数组已经有序

所以第二趟排序我们在这里略过

直接显示第三趟排序

得到partion=2

第三趟排序

Key=6

high=5

得到partion=5

Low=4

1 2 3 4 5 6 7 8

交换

接下来数组已经有序，其它趟数的

排序方法和上面一样。因为数组已

经有序在这里就不在赘述

下面附上快速排序的PHP代码

$array=array(3,5,2,6,8,1,7,4);

function quickSort(&$arr,$low,$high){

if($low<$high){

$l=$low;

$h=$high;

$key=$arr[$low];

while($l<$h){

while($l<$h&&$arr[$h]>$key) $h--;

$arr[$l]=$arr[$h];

while($l<$h&&$arr[$l]<$key) $l++;

$arr[$h]=$arr[$l];

}

$arr[$l]=$key;

quickSort($arr,$low,$l-1);

quickSort($arr,$l+1,$high);

}else{

return ;

}

}

quickSort($array,0,count($array)-1);

最坏的情况 是发生在划分过程中产生的两个区域分别包含n-1个元素和0个元素的时候。时间复杂度为O（n^2）

最快情况下，即PARTITION可能做的最平衡的划分中，得到的每个子问题都不能大于n/2。时间复杂度为O（nlgn）

选择排序

选择排序是一种简单直观的排序算法

算法的基本思想是 在要排序的数组中找出一个最小（或最大）的数与第一个位置的数交换；然后再从剩下的数组中找出在当前数组中最小（或最大）的数与第二个位置的数交换；以此类推，直到第n-1个元素（倒数第二个）和第n个元素（最后一个数）比较为止。

算法步骤：

1）首先在未排序序列中找到最小（大）元素，存放到排序序列的起始位置

2）再从剩余未排序元素中继续寻找最小（大）元素，然后放到已排序序列的末尾。

3）重复第二步，直到所有元素均排序完毕。

初始数组

3 5 2 6 8 1 7 4

第二趟排序

第一趟排序

1 2 5 6 8 3 7 4

1 5 2 6 8 3 7 4

1 2 3 4 5 8 7 6

1 2 3 6 8 5 7 4

1 2 3 4 8 5 7 6

第三趟排序

第四趟排序

第五趟排序

下面附上选择排序的PHP代码

第六趟排序

1 2 3 4 5 6 7 8

$array=array(3,5,2,6,8,1,7,4);

function selectSort(&$arr){

for($i=0;$i<count($arr);$i++){

$store=$arr[$i];

$pos=$i;

$flag=false;

for($j=$i;$j<count($arr);$j++){

if($arr[$j]<$store){

$store=$arr[$j];

$pos=$j;

$flag=true;

}

}

if($flag==true){

$arr[$pos]=$arr[$i];

$arr[$i]=$store;

}

}

}

selectSort($array);

选择排序时间复杂度：O(n^2)

冒泡排序

冒泡排序（Bubble Sort）是一种简单直观的排序算法。它重复地走访过要排序的数列，一次比较两个元素，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，也就是说该数列已经排序完成。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。

算法步骤：

1）比较相邻的元素。如果第一个比第二个大，就交换他们两个。

2）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。这步做完后，最后的元素会是最大的数。

3）针对所有的元素重复以上的步骤，除了最后一个。

4）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。

初始数组

3 5 2 6 8 1 7 4

第八趟冒泡

第七趟冒泡

1 2 3 4 5 6 7 8

1 2 3 4 5 6 7 8

第六趟冒泡

第五趟冒泡

第四趟冒泡

第三趟冒泡

第二趟冒泡

第一趟冒泡

1 2 3 4 5 6 7 8

2 3 5 1 4 6 7 8

2 3 1 4 5 6 7 8

2 1 3 4 5 6 7 8

2 3 5 6 1 4 7 8

3 2 5 6 1 7 4 8

按照上面的排序数组，理论上是有八趟冒泡，但是我们会发现在第六趟冒完泡以后数组已经有序了，所以在实际应用中我们会判断如果数组已经有序了，那么我们就会停止接下来的冒泡。

下面附上冒泡排序的PHP代码

$array=array(3,5,2,6,8,1,7,4);

function bublingSort(&$arr){

for($i=count($arr)-1;$i>0;$i--){

$flag=false;

for($j=0;$j<$i;$j++){

if($arr[$j]>$arr[$j+1]){

$tmp=$arr[$j];

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

$arr[$j+1]=$tmp;

$flag=true;

}

}

if($flag==false){

break;

}

}

}

bublingSort($array);

由以上程序分析可知

冒泡排序的时间复杂度为O(n^2)

归并排序

归并排序用到了分治策略。

用分治策略解决问题分为三步：分解、解决、合并。也即：将原问题划分成n个规模较小而结构与原问题相似的子问题； 递归地解决这些子问题，然后再合并其结果，得到原问题的解。此处n=2

归并过程为：

比较a[i]和a[j]的大小，若a[i]≤a[j]，则将第一个有序表中的元素a[i]复制到r[k]中，并令i和k分别加上1；否则将第二个有序表中的元素a[j]复制到r[k]中，并令j和k分别加上1，如此循环下去，直到其中一个有序表取完，然后再将另一个有序表中剩余的元素复制到r中从下标k到下标t的单元。归并排序的算法我们通常用递归实现，先把待排序区间[s,t]以中点二分，接着把左边子区间排序，再把右边子区间排序，最后把左区间和右区间用一次归并操作合并成有序的区间[s,t]。

归并排序时间复杂度：

时间复杂度为O(nlogn) 这是该算法中最好、最坏和平均的时间性能。

空间复杂度为 O(n)

比较操作的次数介于(nlogn) / 2和nlogn - n + 1。

赋值操作的次数是(2nlogn)。归并算法的空间复杂度为：0 (n)

归并排序比较占用内存，但却是一种效率高且稳定的算法。

排序图示：

初始数组

3 5 2 6 8 1 7 8 4

第一趟

第二趟

第三趟

第五趟

{1 2 3 4 5 6 7 8 8}

第四趟

{1 2 3 5 6 7 8 8} {4}

{2 3 5 6} {1 7 8 8} {4}

{3} {5} {2} {6} {8} {1} {7} {8} {4}

{3 5} {2 6} {1 8} {7 8} {4}

由上图可以看出，归并排序算法是稳定的排序算法

下面附上归并排序算法代码：（1）

$array=array(3,5,2,6,8,1,7,8,4);

function merge($lArr,$rArr){

$arr=array();

while(count($lArr)>0&&count($rArr)>0){

$arr[]=$lArr[0]<=$rArr[0]?array\_shift($lArr):array\_shift($rArr);

}

return array\_merge($arr,$lArr,$rArr);

}

function mergeSort($arr){

$arrlen=count($arr);

if($arrlen<=1) return $arr;

$mid=intval($arrlen/2);

$lArr=array\_slice($arr,0,$mid);

$rArr=array\_slice($arr,$mid);

$lArr=mergeSort($lArr);

$rArr=mergeSort($rArr);

$arr=merge($lArr,$rArr);

return $arr;

}

$array=mergeSort($array);

由于php代码对于数组的处理非常的方便，所以使用php编写归并排序算法很简便。

但是为了更好的理解归并排序算法，所以小编在用php模拟C语言的方式来编写归并排序算法代码。

下面是归并排序代码 （2）

function merge(&$arr,&$tarr,$starpos,$mid,$endpos){

$l=$starpos;

$r=$mid+1;

$k=$starpos;

while($l<=$mid&&$r<=$endpos){

if($arr[$l]<$arr[$r]){

$tarr[$k++]=$arr[$l++];

}else{

$tarr[$k++]=$arr[$r++];

}

}

while($l<=$mid){

$tarr[$k++]=$arr[$l++];

}

while($r<=$endpos){

$tarr[$k++]=$arr[$r++];

}

for($i=$starpos;$i<=$endpos;$i++){

$arr[$i]=$tarr[$i];

}

}

function mergeSort(&$arr,&$temparr,$startpos,$endpos){

if($startpos<$endpos){

$mid=intval(($startpos+$endpos)/2);

mergeSort($arr,$temparr,$startpos,$mid);

mergeSort($arr,$temparr,$mid+1,$endpos);

merge($arr,$temparr,$startpos,$mid,$endpos);

}

}

$temparr=array();

mergeSort($array,$temparr,0,count($array)-1);