# Csharp\_八大排序

一、定义与前言

**1. 排序:是将一个数据元素（或记录）的任意序列，重新排列成一个按关键字有序的序列。**

**2.** 排序根据涉及的存储器的不同分为内部排序和外部排序：

a) 内部排序: 指待排序记录存放在内存进行的排序过程；

b) 外部排序: 指待排序记录的数量很大，以致内存一次不能容纳全部记录，在排序过程中尚需对外存进行访问的排序过程。

**本文仅讨论内部排序,以下是关系图：**



一、插入排序（重在插入，挪动元素）

1. 直接插入排序
2. 基本思想：将一个记录插入到已排序好的有序表中，从而得到一个新的有序表。即：先将序列的第1个记录看成是一个有序的子序列，然后从第2个记录逐个进行插入，直至整个序列有序为止。
3. 要点：设立哨兵，作为临时存储和判断数组边界之用。
4. 直接排序示例：

手机屏幕截图

描述已自动生成

1. 代码示例：[见\_1StraightInsertSort.cs](_1StraightInsertSort.cs)
2. 流程分析：
   1. 将要排序的数（索引为i）存储起来，向前查找合适位置j+1，将i-1到j+1的元素依次向后
   2. 移动一位，空出j+1，然后将之前存储的值放在这个位置。
   3. 这个方法写的不如维基上的简洁清晰，由于合适位置是j+1所以多出了对j==0的判断，但实际效率影响无差别。
3. 算法时间复杂度：直接插入排序（o(n2)）

2. 二分法直接插入排序

1. 说明：插入排序主要工作是在有序的数列中对要排序的数查找合适的位置，而查找里面经典的二分查找法正可以适用。
2. 原理：通过二分查找法的方式找到一个位置索引。当要排序的数插入这个位置时，大于前一个数，小于后一个数。
3. 图示：

手机屏幕截图

描述已自动生成

1. 算法时间复杂度：二分法插入排序（o(nlogn)）
2. 代码示例：\_[2InsertSortWithBinarySearch](_2InsertSortWithBinarySearch.cs).cs

3. 希尔排序（shellSort）

1. 说明：希尔排序是1959年由D.L.Shell 提出来的，相对直接排序有较大的改进，又称“缩小增量排序”。
2. 思想：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“基本有序”时，再对全体记录进行依次直接插入排序。
3. 图示：一些文字和图片的手机截图

   描述已自动生成
4. 操作方法：
   1. 选择一个增量序列t1，t2，…，tk，其中ti>tj，tk=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ti，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。
5. 算法时间复杂度：希尔排序最好（o(nlogn)），最坏（o(n2)）.
6. 代码示例：[\_\_3ShellSort.cs](_3ShellSort.cs)

二、选择排序（重在比较）

1. 简单选择排序

1. 说明：类似于打擂台法。
2. 基本思想：在要排序的一组数中，选出最小（或者最大）的一个数与第1个位置的数交换；然后在剩下的数当中再找最小（或者最大）的与第2个位置的数交换，依次类推，直到第n-1个元素（倒数第二个数）和第n个元素（最后一个数）比较为止。
3. 图示：图片包含 游戏机

   描述已自动生成
4. 操作方法：
   1. 第一趟，从n 个记录中找出关键码最小的记录与第一个记录交换；
   2. 第二趟，从第二个记录开始的n-1 个记录中再选出关键码最小的记录与第二个记录交换；
   3. 以此类推.....
   4. 第i 趟，则从第i 个记录开始的n-i+1 个记录中选出关键码最小的记录与第i 个记录交换，直到整个序列按关键码有序。
5. 算法时间复杂度：（o(n2)）
6. 代码示例：<_4SimpleSelectSort.cs>

2. 二元选择排序

1. 说明：简单选择排序的改进。
2. 基本思想：在要排序的一组数中，选出最小和最大的两个数放置于数组被排序的数的两端
3. 图示：屏幕上写着字

   描述已自动生成
4. 操作方法：
   1. 第一趟，从n 个记录中找出关键码最小和最小记录并分别放入未排序范围的两边
   2. 第二趟，从第二个记录开始的n-1 个记录中再选出关键码最小和最小记录并分别放入未排序范围的两边。
   3. 以此类推.....
   4. 第i 趟，从第i个记录开始的n-i 个记录中再选出关键码最小和最小记录并分别放入未排序范围的两边。直到整个序列按关键码有序。
5. 算法时间复杂度：（o(n2)）
6. 代码示例：<_5DualisticSelectSort.cs>

三、交换排序

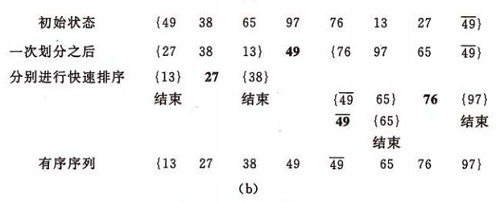
1.冒泡排序

1. 原理：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。
2. 图示：手机屏幕截图

   描述已自动生成
3. 改进1：加入一标志性变量exchange，用于标志某一趟排序过程中是否有数据交换，如果进行某一趟排序时并没有进行数据交换，则说明数据已经按要求排列好，可立即结束排序，避免不必要的比较过程。代码见：<_6BubbleSort.cs>
4. 改进2：进行同时的正向和反向冒泡一次得到两个最终值，从而减少一半排序的趟数。代码见：<_7PingPongBubbleSort.cs>
5. 算法时间复杂度：（o(n2)）

2.快速排序

1. 原理：
   1. 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,
   2. 通过一趟排序讲待排序的记录分割成独立的两部分，其中一部分记录的元素值均比基准元素值小。另一部分记录的 元素值比基准值大。
   3. 此时基准元素在其排好序后的正确位置
   4. 然后分别对这两部分记录用同样的方法继续进行排序，直到整个序列有序。
2. 图示：
   1. 一趟排序的过程：图片包含 游戏机

      描述已自动生成
   2. 排序的全过程：
3. 算法时间复杂度：（o(nlogn)）
4. 改进前的代码见：<_8QuickSort.cs>
5. 改进思路：交换排序当交换的数组元素过少也递归的话，会造成创建内存函数的时间指数增加并超过交换元素的时间，所以基于快排一定的顺序性进行，代码见：<_9QuickInsertSort.cs>