

**冒泡：**

每次排序后最大的元素浮到最后。

基本不用，太慢。需要相邻两两比较两两交换。

**选择：**

第一个位置选择最小的元素，第n个位置选择第n小的。

基本不用，不稳。[A:80、B:80、C:70]，从小到大排序，第一步C和A做交换就完成排序了，但A和B顺序改变了。

**插入：**

将数组分两个空间：已排序、未排序。依次取一个未排序元素插入到已排序空间挪动出的位置。

样本小且基本有序时的效率比较高。

插入比冒泡好，交换两个元素时冒泡需要3次赋值操作，插入排序仅需1次赋值（已排序元素简单挪到后一个位置）。

[选择排序为什么不稳定\_面试官：说说插入、冒泡、选择排序？哪个更好？为什么？...\_环形剧场现任负责人的博客-CSDN博客](https://blog.csdn.net/weixin_36003005/article/details/113411025)

**希尔排序：**

间隔n..1分组跳排几次，最后1意味着全部插入排序。希尔排序是插入排序的优化版，间隔大时移动次数少（末尾小元素移到头部所需次数减少）、间隔小时移动距离短。

间隔跳排会导致不稳定。

[图解排序算法(二)之希尔排序 - dreamcatcher-cx - 博客园 (cnblogs.com)](https://www.cnblogs.com/chengxiao/p/6104371.html)

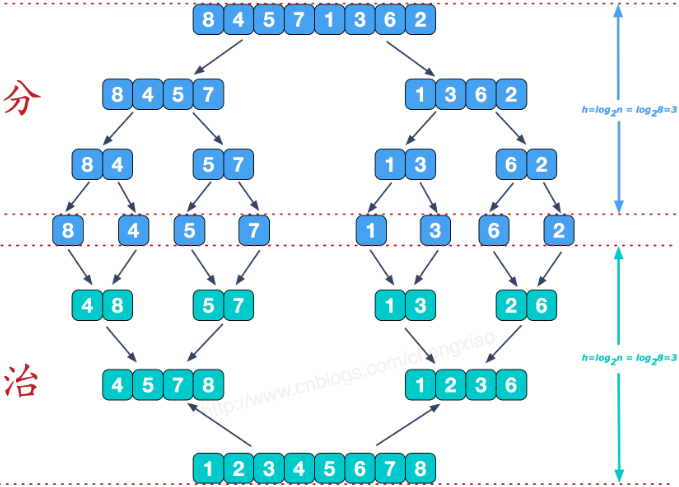
**归并排序（分治）：**

假设得到两个有序序列[4,5,7,8]和[1,2,3,6]，那么只需简单地顺序各取一个元素比较，大的元素继续与另一个序列取出的元素比较，即可排列成一个有序序列。

从以上知可对一个序列进行分治，先分解为两个子序列各自排好序，再归并成一个有序序列，不断重复分治子序列。

有两种实现方式：自顶向下的递归、自底向上的迭代。但本质都一样，先排好小序列最后排大序列。

[归并排序 | 菜鸟教程 (runoob.com)](https://www.runoob.com/w3cnote/merge-sort.html)



**TimSort：**

归并排序的改进版，使用了归并+二分插入排序。

**快速排序（分治）：**

与归并排序相反，快排是先对大序列排好范围序（小于pivot、pivot、大于pivot），再不断分治对子序列快排。、

不适合对已排好序的序列快排，时间复杂度是最坏的O(N\*N)，因为每一轮快排只能得到一个轴数和一个剩余序列（退化为冒泡排序），需要遍历N轮，不是最好的logN。

原地排序的空间复杂度是O(1)，真正消耗空间的是递归调用的栈空间，最优、最差空间复杂度如下：

最优情况下空间复杂度为O(logN)，每一轮都平分数组；

最差情况下空间复杂度为O(N)，退化为冒泡排序，每一轮取到的轴数是该序列中最大/最小的，即每一轮只排好一个元素需要N轮才能排完；

基准选择：

不同的基准不影响排序结果，但影响整体排序时间。不同基准分割出的左右两部分不同，如果左右集合几乎相等，那么整体分割的次数显然会减少，整体耗时也相应降低。

1. 选择第一个或者最后一个：如果待排序序列是有序的，这将产生糟糕的分割，时间复杂度是最差的O(N\*N)；
2. 随机选择：序列中随即选出一个元素作为基准；
3. 三数中值：选择最左端、最右端和中间位置的三个元素的中值作为基准。

由于比较和交换是跳跃的，因此快排不稳定。

[快速排序你真的会了吗？ - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/57436476)

[快速排序的稳定化实现\_liuchenjane的博客-CSDN博客\_稳定快速排序](https://blog.csdn.net/liuchenjane/article/details/72902325)

[一篇文章让你真正了解快速排序 - SegmentFault 思否](https://segmentfault.com/a/1190000017314698)

[快速排序 及其时间复杂度和空间复杂度\_一诺-CSDN博客\_快速排序复杂度](https://blog.csdn.net/A_BlackMoon/article/details/81064712)

**堆排序：**

堆是一种CBT/完全二叉树。堆是弱序的（只保证根到叶的每一条路径内有序），不支持直接按序遍历，但可以快速移除根节点、插入新节点。

堆排序流程：

1. 先将数组从尾到头逐个进行堆化调整，时间复杂度O(N)；
2. 再对数组进行N次堆头弹出到堆尾，时间复杂度O(N\*logN)。

总时间复杂度O(N + N\*logN)。

[堆-树（完全二叉树） - 大华1100 - 博客园 (cnblogs.com)](https://www.cnblogs.com/zhoanghua/p/9288899.html)

**计数排序（桶、非比较、多关键字）：**

使用一个桶映射不同数值出现的次数，再按序根据次数依次排列。

适合量大但范围小的序列。

时间复杂度：遍历一遍原序列并记录到累加序列，累加一遍计数序列，最后倒序排放一遍，2(N+K)约等于O(N+K)。

空间复杂度：计数序列、结果序列，等于O(N+K)。

[一文弄懂计数排序算法！ - 程序员小川 - 博客园 (cnblogs.com)](https://www.cnblogs.com/xiaochuan94/p/11198610.html)

**基数排序（桶、非比较、多关键字）：**

以数值的位数为关键字，使用一个桶映射该位数值出现的次数，完成多轮按序排列。

时间复杂度：与位数的数量K成正比，约等于O(N\*K)。

有两种排法：LSD低位优先、MSD高位优先（属于分治）。

**桶排序：**

  
需划分区间，有可能全部元素都位于一个区间，所以每个桶都得保证有完整空间来存放，这将导致空间冗余；

空间要做到最好的话，就只能用链表来存放，但时间就做不到最好了。

时间复杂度：

最好情况是每个桶只有一个元素，O(N)；

最坏情况是全部元素分到一个桶上，该桶的最差排序可能是O(N\*N)；

平均为”遍历求最大最小值N+桶初始化K+遍历装桶N+桶内排序(N/K)\*log(N/K)\*K+遍历桶输出N“约等于N+K。

实际中，桶排序用的少，计数排序和基数排序用的多；

一般来说，基数排序要求，样本是整数，且范围比较窄；

一般来说，基数排序要求，样本是10进制的正整数；

一旦要求稍有升级，相对于基于比较排序，改写代价增加是显而易见的。

**总结：**

1. 不基于比较的排序，对样本数据有严格要求，不易改写；
2. 基于比较的排序，只要规定好两个样本怎么比较大小，就可以直接复用；
3. 基于比较的排序，时间复杂度的极限是O(N\*logN)；
4. 大数据量时为了绝对的速度选快排（O(N\*logN)、O(logN））、为了省空间选堆排（O(N\*logN)、1）、为了稳定性选归并（O(N\*logN)、N）；
5. 时间复杂度低于O(N\*logN)、额外空间复杂度低于O(N)、且稳定，基于比较的排序是不存在的！