排序算法：

1.内部排序：

1. 插入排序：直接插入排序，希尔排序
2. 选择排序：简单选择排序，堆排序
3. 交换排序：冒泡排序，快速排序
4. 归并排序

2.外部排序

（排序的稳定性分析，就根据代码来判断吧，毕竟写了个<=和<是不一样效果）

1. **直接插入排序（摸扑克牌并且理牌的场景）**

**基本思想**：将一个记录插入到已经有序的序列中

**时间复杂度**：O(n^2)；当数组近乎有序的时候，复杂度近乎O(n)

**大概的样子**：数据一边是有序的，但是不能保证这部分是整个数组中最小/大

**改进**：

思路1：设置flag位置，如果本轮没发生交换，那么提前停止排序；再用直接赋值的方法最后一次完成交换(该方法不适用交换策略)。

那么优化后，对于有序的数组，性能很高啊

1. **希尔排序**

**基本思路：**将整个元素序列基于增量分成若干子序列，对于子序列进行直接插入排序；一次缩小增量进行排序

**时间复杂度：**O(n^(3/2))

**大概的样子：**不可描述

1. **冒泡排序**

**基本思想**：两两比较相邻位置的记录，如果反序则交换

**时间复杂度**：O(n^2)；最优(基本有序)：O(n)

**大概的样子**：数组一边(根据大小排序具体分析)有序，而且是后面的排序中该部分不在变化，有序的部门逐渐变大

代码：

**改进**：

思路1：设置flag位置，如果排序一趟没有发生交换，那么提前停止排序

1. **简单选择排序**

**基本思路：**(可能有不同的思路哦)寻找(i,n)区间的最小值,和区间左臂的前一个元素交换

**时间复杂度：**O(n^2),但一般来说简单选择排序的性能要优于冒泡排序

**大概的样子：**最小的先排出来，然后第二小，然后…

1. **堆排序**

**基本思路：**堆是具有下列性质的完全二叉树:每个节点的值都大于或等于其左右孩子节点的值,称为大顶堆；或者每个节点的值都小于或等于其左。将待排序的序列构造成一个大顶堆.此时,整个序列的最大值就是堆顶 的根结点.将它移走(其实就是将其与堆数组的末尾元素交换, 此时末尾元素就是最大值),然后将剩余的n-1个序列重新构造成一个堆,这样就会得到n个元素的次大值.如此反复执行,便能得到一个有序序列了。

**时间复杂度：O(**nlogn**)**

**大概的样子：**不可描述

1. **快速排序**

**基本思想：**通过一趟排序找，基于第一个元素，将小于该元素的放在前面，否则就放在后面。得到前后两个数组，分别对这两个数组用同样的方法分裂，递归下去。

**时间复杂度：**O(nlogn),接近有序的数组最差复杂度：O(n^2),but发生这样的情况的概率极低

**大概的样子：**有序的几个部分

**改进：**

**思路1（不太显著的优化）：**在数组较小的时候，采用优化的插入排序(因为递归到数组较小的时候，有一定的概率接近有序的数组)

**思路2：**针对近乎有序的数组，在选择基点元素的时候随机选择

**思路3：**针对接近有序，有大量重复的数组，三路快排

1. **归并排序**

**基本思路：**不断的二分递归，得到最少元素集，对最少元素集按照大小合并两个数组，不断的递归回去

**时间复杂度：**O(nlogn)

**注：**如果使用非递归的方式，可以对链表这种数据结构进行排序