**2.1 初级排序算法**

**2.1.1 游戏规则**

编程：排序算法类的模版

Less，exch，show，issorted方法

排序算法的额外内存开销和运行时间是同等重要的。

不需要额外内存的称为原地排序算法。

**2.1.2 选择排序**

首先，找到数组中最小的那个元素，其次，将它和数组的第一个元素交换位置。再次，在剩下的元素中找到最小的元素，如此往复。

N的平方/2次比较和N次交换。

编程：选择排序

**2.1.3 插入排序**

通常人们整理桥牌的方法是一张一张的来，将每一张牌插入到其他已经有序的牌中的适当位置。

为了给要插入的元素腾出空间，我们需要将其余所有元素在插入之前都向右移动一位。

对于随机排列的长度为N且主键不重复的数组，平均情况下插入排序需要N2/4次比较和交互。最坏情况N2/2次比较和交换。最好情况需要N-1次比较，0次交换。

事实上，当倒置的数量很少时，插入排序很可能比本章中的其他任何算法都要快。

插入排序对于部分有序的数组十分高效，也很适合小规模数组。而且它也是高级排序算法的中间过程。

编程：插入排序

**2.1.4 排序算法的可视化**

**2.1.5 比较两种排序算法**

**2.1.6 希尔排序**

希尔排序为了加快速度简单地改进了插入排序，交换不相邻的元素以对数组的局部进行排序，并最终用插入排序将局部有序的数组进行排序。

希尔排序的思想是使数组中任意间隔为h的元素都是有序的。这样的数组称为h有序数组。

透彻了解希尔排序的性能至今仍然是一项挑战。

编程：希尔排序

和选择排序以及插入排序形成对比的是，希尔排序可以用于大型数组。

希尔算法在最坏的情况下算法2.3比较次数和N的3/2次方成正比。

对于中等大小的数组，希尔排序的运行时间是可以接受的。它的代码量很小，且不需要外的内存空间。