插入排序

插入排序算法是所有排序方法中最简单的一种算法，其主要的实现思想是将数据按照一定的顺序一个一个的插入到有序的表中，最终得到的序列就是已经排序好的数据。

1. 直接插入排序

直接插入排序是插入排序算法中的一种，采用的方法是：在添加新的记录时，使用顺序查找的方式找到其要插入的位置，然后将新记录插入。

直接插入排序(Straight Insertion Sort)的基本思想是：

· 把n个待排序的元素看成为一个有序表和一个无序表。

· 开始时有序表中只包含1个元素，无序表中包含有n−1个元素，排序过程中每次从无序表中取出第一个元素，将它插入到有序表中的适当位置，使之成为新的有序表，重复n−1次可完成排序过程。

> 插入排序算法还包括**折半插入排序**、**2−路插入排序**和**希尔排序**等

例如采用直接插入排序算法将无序表{3,1,7,5,2,4,9,6}进行升序排序的过程为：

- 首先考虑记录 3 ，由于插入排序刚开始，有序表中没有任何记录，所以 3 可以直接添加到有序表中，则有序表和无序表可以如图 1所示：

- 向有序表中插入记录 1 时，同有序表中记录 3 进行比较，1<3，所以插入到记录 3 的左侧，如图所示：

- 向有序表插入记录 7 时，同有序表中记录 3 进行比较，3<7，所以插入到记录 3 的右侧，如图 所示：

- 向有序表插入记录 2 时，同有序表中记录 7进行比较，2<7，再同 5，3，1分别进行比较，最终确定 2 位于 1 和 3 中间，如图所示：

- 照此规律，依次将无序表中的记录 4，9 和 6插入到有序表中，如图所示：

(1)插入 4

(2)插入 9

(3)插入 6

直接插入排序：将待排序的数组，分成两个序列，前面的序列保持有序，依次选择后面的元素，往前面插入。（就地排序、稳定排序）

根据数据量进行一个划分，C++的STL当数据少了就用插入排序（<=8个）

2. 折半插入排序

上述算法在查找插入位置时，采用的是顺序查找的方式，而在查找表中数据本身有序的前提下，可以使用折半查找来代替顺序查找，这种排序的算法就是折半插入排序算法。

3. 2–路插入排序

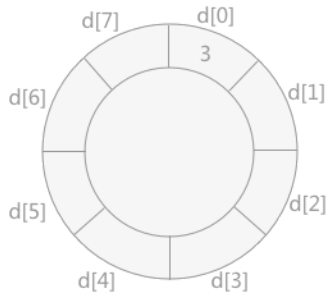
2–路插入排序算法是在折半插入排序的基础上对其进行改进，减少其在排序过程中移动记录的次数从而提高效率。

具体实现思路为：另外设置一个同存储记录的数组大小相同的数组 d，将无序表中第一个记录添加进 d[0] 的位置上，然后从无序表中第二个记录开始，同 d[0] 作比较：如果该值比 d[0] 大，则添加到其右侧；反之添加到其左侧。

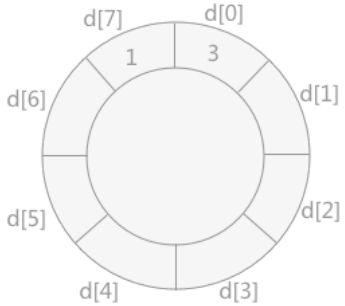
> 在这里的数组 d 可以理解成一个环状数组。

例如：使用 2–路插入排序算法对无序表{3,1,7,5,2,4,9,6}排序的过程如下：

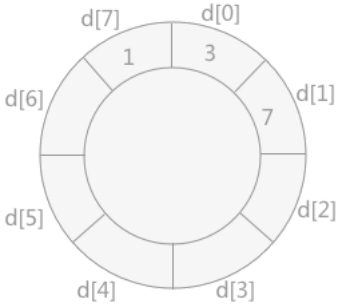
- 将记录 3 添加到数组 d 中：



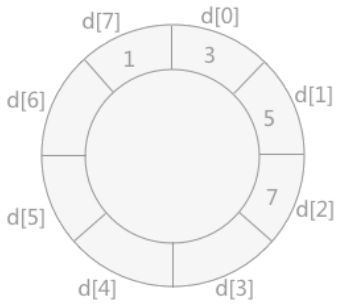
- 然后将 1 插入到数组 d 中，如下图所示：



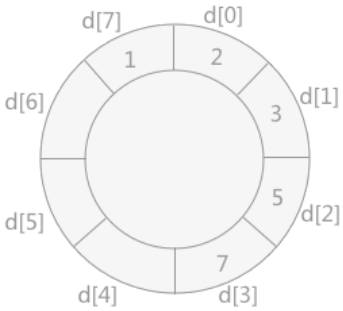
- 将记录 7 插入到数组 d 中，如下图所示：



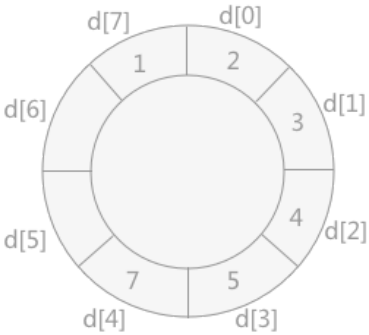
- 将记录 5 插入到数组 d 中，由于其比 7小，但是比 3 大，所以需要移动 7 的位置，然后将 5 插入，如下图所示：



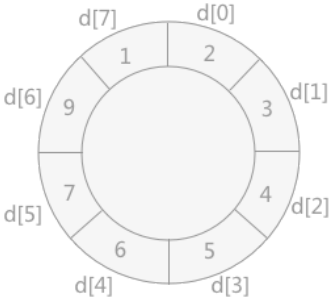
- 将记录 2 插入到数组 d 中，由于比 1大，比 3 小，所以需要移动 3、7、5 的位置，然后将 2 插入，如下图所示：



- 将记录 4 插入到数组 d 中，需要移动 5 和 7 的位置，如下图所示：



- 将记录 9 插入到数组 d 中，将记录 6 插入到数组 d 中，如下图所示：



3. 希尔排序

希尔排序，又称“缩小增量排序”，也是插入排序的一种，但是同前面几种排序算法比较来看，希尔排序在时间效率上有很大的改进。

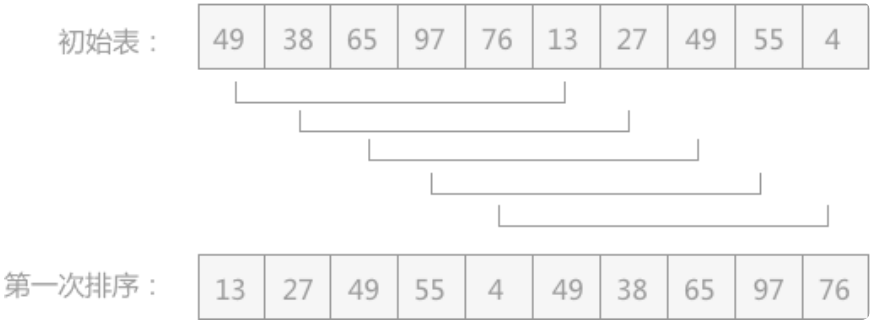
在使用直接插入排序算法时，如果表中的记录只有个别的是无序的，多数保持有序，这种情况下算法的效率也会比较高；除此之外，如果需要排序的记录总量很少，该算法的效率同样会很高。希尔排序就是从这两点出发对算法进行改进得到的排序算法。

> 希尔排序的具体实现思路是：先将整个记录表分割成若干部分，分别进行直接插入排序，然后再对整个记录表进行一次直接插入排序。

比如 8 9 1 7 2 3 5 4 6 0

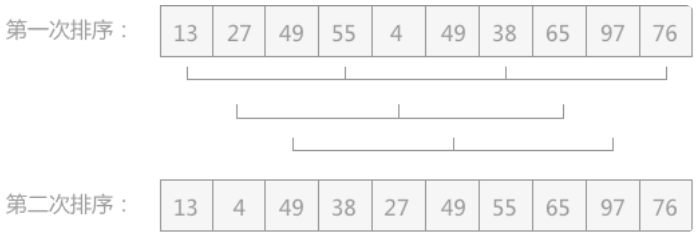
例如无序表{49，38，65，97，76，13，27，49，55，4}进行希尔排序的过程为：

- 首先对 {49，13}，{38，27}，{65，49}，{97，55}，{76，4} 分别进行直接插入排序（如果需要调换位置也只是互换存储位置），如下图所示：



上图中两两进行比较，例如 49 和 13 进行比较，13<49，所以交换存储位置。

- 通过一次排序，无序表中的记录已基本有序，此时还可以再进行一次分割，如下图所示：



- 经过两次分割，无序表中已基本有序，此时对整张表进行一次直接插入排序（只需要做少量的比较和插入操作即可），最终希尔排序的结果为:

希尔排序的过程中，对于分割的每个子表，其各自包含的记录在原表中并不是相互挨着的，而是相互之间相隔着某个固定的常数。例如上例中第一次排序时子表中的记录分割的常量为 5，第二次排序时为 3。

通过此种方式，对于关键字的值较小的记录，其前移的过程不是一步一步的，而是跳跃性的前移，并且在最后一次对整表进行插入排序时减少了比较和排序的次数。