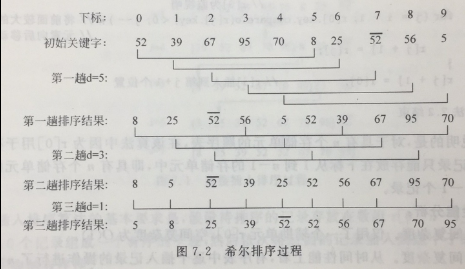
直接插入排序算法简单，在n值比价小时，效率比较高；当n很大的时候，若待排序序列时按照关键字基本有序，效率依然很高，其时间效率可提高到O(n)。**希尔排序正是从这两点出发给出的插入排序的改进方法。**

**希尔排序又称缩小增量排序。**

**基本思路**

先去一个小于n的整数 di (称为增量)，然后把排序表中的n个记录分为 di 个子表，从下标为0的记录开始，间隔为di 的记录组成一个子表，在各个子表内进行直接插入排序。在一趟之后，间隔为di 的记录组成的子表已有序，随着有序性的改善，逐步减少增量di ，重复上述操作，直到di=1，使得间隔为1的记录有序，也就是整个序列都达到了有序。



图中第一趟di=5，所以从0开始，0和5为一组，1和6为一组，… ,4和9为一组。

然后进行直接插入排序。

依次类推。**注意，最后一个增量必须为1.**

**步骤**

1.选择一个增量序列{d0，d1，…，di}

2.根据当前增量di将n条记录分成di个子表，每个子表中记录的下标相隔为di

3.对各个子表中的记录进行直接插入排序

4.重复，选择下一个增量进行下一趟排序

**代码**

public static int[] shellSort(int[] d,int[] nums){ //d为增量序列，nums为待排序序列

for(int k=0;k<d.length;k++) { //取每个增量dk进行插入排序

int dk = d[k];

for (int c=0; c < dk; c++) { //第c个子表，每个子表的起点为c+0，第二个数据为c+dk

for (int i = dk+c; i < nums.length; i = i + dk) { //注意，i起点变成了dk+c，而不是0+c.因为之前增量为1，现在为dk

//i为当前子表的第二个元素，因为第一个元素不用比较直接入有序序列，所以跳过直接开始第二个

int temp = nums[i]; //对当前i进行插入排序

int j = i - dk; //i的前一位是i-dk，j=i-dk

while (j >= 0 && nums[j] > temp) { //判断下标是否有效j>=0,同时判断每个的关键字大小

nums[j + dk] = nums[j];//大于就向后移动一位，这里增量为dk，所以向后移动dk位

j = j - dk;//上一位数据位置为j-dk，从后往前

}

nums[j + dk] = temp;

}

}

}

return nums;

}

**算法性能分析**

1.空间复杂度

由于希尔排序用到了直接插入排序，所以空间复杂度同为O(1)

2.时间复杂度

希尔排序的时间效率分析很困难，关键字的比较次数与记录的移动次数依赖与增量序列的选取，在特定情况下可以准确估算出关键字的比较次数和记录的移动次数。一些好的增量序列可以很**接近O()**

**稳定性：不稳定**