阅读：4,360　　作者：解学武

希尔排序算法（缩小增量排序）及C语言实现

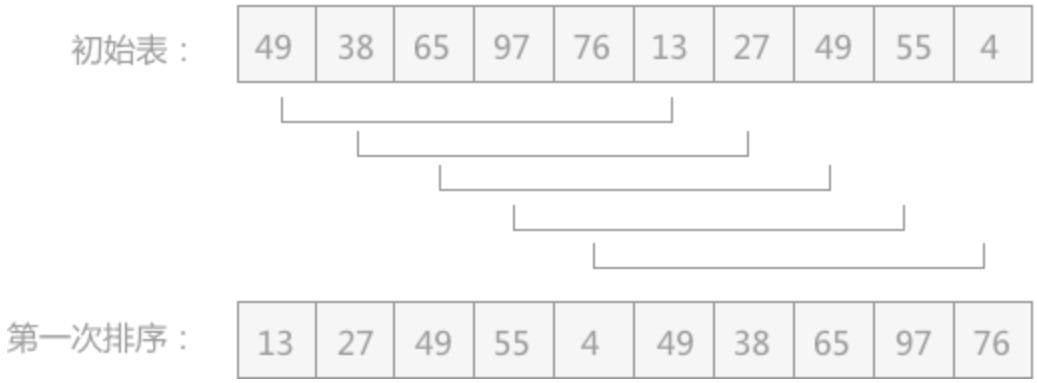
希尔排序，又称“缩小增量排序”，也是插入排序的一种，但是同前面几种排序算法比较来看，希尔排序在时间效率上有很大的改进。

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。

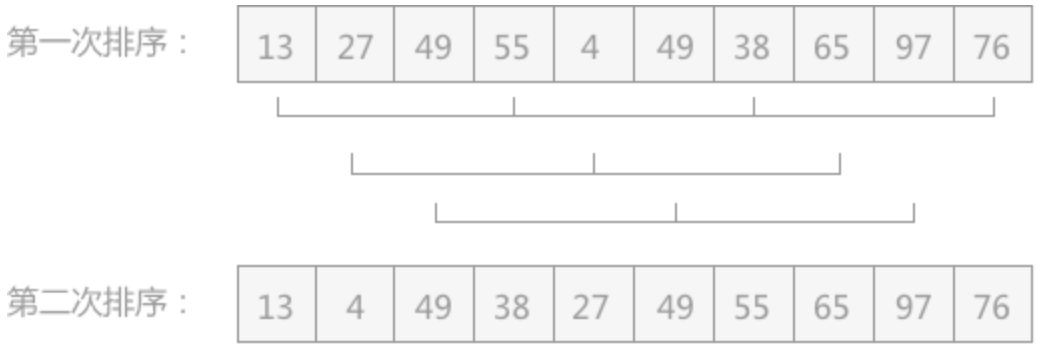
例如无序表 {49，38，65，97，76，13，27，49，55，4} 进行希尔排序的过程为：

- 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：

最后一次排序：	4	13	27	38	49	49	55	65	76	97

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。

> 一般在记录的数量多的情况下，希尔排序的排序效率较直接插入排序高。

希尔排序的具体代码实现：

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 15
typedef struct {
    int key;    //关键字的值
}SLNode;
typedef struct {
    SLNode r[SIZE];//存储记录的数组
    int length;//记录数组中记录的数量
}SqList;
//希尔排序的实现函数，其中 dk 表示增值量
void ShellInsert(SqList * L,int dk){
    //从 dk+1 个记录起，每间隔 dk 个记录就调取一个进行直接插入排序算法
    for (int i=dk+1; i<=L->length; i++) {
        //如果新调取的关键字的值，比子表中最后一个记录的关键字小，说明需要将该值调换位置
        if (L->r[i].key<L->r[i-dk].key) {
            int j;
            //记录表中，使用位置下标为 0 的空间存储需要调换位置的记录的值
            L->r[0]=L->r[i];
            //做直接插入排序，如果下标为 0 的关键字比下标为 j 的关键字小，则向后一行下标为 j 的值，为
            for (j=i-dk; j>0 && (L->r[0].key<L->r[j].key); j-=dk){
                L->r[j+dk]=L->r[j];
            }
            //跳出循环后，将新的记录值插入到腾出的空间中，即完成了一次直接插入排序
            L->r[j+dk]=L->r[0];
        }
    }
}
//希尔排序，通过调用不同的增量值（记录），实现对多个子表分别进行直接插入排序
void ShellSort(SqList * L,int dlta[],int t){
    for (int k=0; k<t; k++) {
        ShellInsert(L, dlta[k]);
```

```
    }
}
int main(int argc, const char * argv[]) {
    int dlta[3]={5,3,1};//用数组来存储增量值，例如 5 代表每间隔5个记录组成一个子表，1表示每间隔一个，
    SqList *L=(SqList*)malloc(sizeof(SqList));
    L->r[1].key=49;
    L->r[2].key=38;
    L->r[3].key=64;
    L->r[4].key=97;
    L->r[5].key=76;
    L->r[6].key=13;
    L->r[7].key=27;
    L->r[8].key=49;
    L->r[9].key=55;
    L->r[10].key=4;
    L->length=10;
    //调用希尔排序算法
    ShellSort(L, dlta, 3);
    //输出语句
    for (int i=1; i<=10; i++) {
        printf("%d ",L->r[i].key);
    }
    return 0;
}
```

运行结果：

```
4 13 27 38 49 49 55 64 76 97
```

> 提示：经过大量的研究表明，所选取的增量值最好是没有除 1 之外的公因子，同时整个增量数组中最后一个增量值必须等于 1，因为最后必须对整张表做一次直接插入排序算法。