希尔排序的实现

希尔排序的实质就是分组插入排序，该方法又称缩小增量排序，因DL．Shell于1959年提出而得名。

该算法的基本思想是:先将整个待排元素序列分割成若干个子序列（由相隔某个“增量”的元素组成的）分别进行直接插入排序，然后依次缩减增量再进行排序，待整个序列中的元素基本有序（增量足够小）时，再对全体元素进行一次直接插入排序。因为直接插入排序在元素基本有序的情况下（接近最好情况），效率是很高的，因此希尔排序在时间效率上比前两种方法有较大提高。

以的一个数组为例:

**第一次 gap = 10 / 2 = 5**

49 38 65 97 26 13 27 49 55 4

1A 2A 3A 4A 5A 1B 2B 3B 4B 5B

1A,1B，2A,2B等为分组标记，数字相同的表示在同一组，大写字母表示是该组的第几个元素， 每次对同一组的数据进行直接插入排序。即分成了五组(49, 13) (38, 27) (65, 49) (97, 55) (26, 4)这样每组排序后就变成了(13, 49) (27, 38) (49, 65) (55, 97) (4, 26)，下同。

**第二次 gap = 5 / 2 = 2**

排序后

13 27 49 55 4 49 38 65 97 26

1A 2A 1B 2B 1C 2C 1D 2D 1E 2E

**第三次 gap = 2 / 2 = 1**

4 26 13 27 38 49 49 55 97 65

1A 1B 1C 1D 1E 1F 1G 1H 1I 1J

**第四次 gap = 1 / 2 = 0 排序完成得到数组：**

4 13 26 27 38 49 49 55 65 97

下面给出严格按照定义来写的希尔排序:

void shellsort1**(**int a**[],** int n**)**

**{**

int i**,** j**,** gap**;**

**for(**gap **=** n **/** 2**;** gap **>** 0**;** gap **/=** 2**)**

**for(**i **=** 0**;** i **<** gap**;** i**++)**

**{**

**for(**j **=** i **+** gap**;** j **<** n**;** j **+=** gap**)**

**if(**a**[**j**]** **<** a**[**j **-** gap**])**

**{**

int temp **=** a**[**j**];**

int k **=** j **-** gap**;**

**while(**k **>=** 0 **&&** a**[**k**]** **>** temp**)**

**{**

a**[**k **+** gap**]** **=** a**[**k**];**

k **-=** gap**;**

**}**

a**[**k **+** gap**]** **=** temp**;**

**}**

**}**

**}**

对上面的shellsort1进行改进和优化。以第二次排序为例，原来是每次从1A到1E，从2A到2E，可以改成从1B开始，先和1A比较，然后取2B与2A比较，再取1C与前面自己组内的数据比较…….。这种每次从数组第gap个元素开始，每个元素与自己组内的数据进行直接插入排序显然也是正确的。

void shellsort2**(**int a**[],** int n**)**

**{**

int j**,** gap**;**

**for(**gap **=** n **/** 2**;** gap **>** 0**;** gap **/=** 2**)**

**for(**j **=** gap**;** j **<** n**;** j**++)**

**if(**a**[**j**]** **<** a**[**j **-** gap**])**

**{**

int temp **=** a**[**j**];**

int k **=** j **-** gap**;**

**while(**k **>=** 0 **&&** a**[**k**]** **>** temp**)**

**{**

a**[**k **+** gap**]** **=** a**[**k**];**

k **-=** gap**;**

**}**

a**[**k **+** gap**]** **=** temp**;**

**}**

**}**

再用直接插入排序的第三种方法来改写下：

void shellsort3**(**int a**[],** int n**)**

**{**

int i**,** j**,** gap**;**

**for(**gap **=** n **/** 2**;** gap **>** 0**;** gap **/=** 2**)**

**for(**i **=** gap**;** i **<** n**;** i**++)**

**for(**j **=** i **-** gap**;** j **>=** 0 **&&** a**[**j**]** **>** a**[**j **+** gap**];** j **-=** gap**)**

swap**(**a**[**j**],** a**[**j **+** gap**]);**

**}**

该步长下算法时间复杂度为。

其他步长：

|  |  |
| --- | --- |
| 步长序列 | 最坏情况下复杂度 |
|  |  |
|  |  |
|  |  |