排序

1. 定义：按数据元素的关键字大小来对数据元素进行有序排列
2. 排序算法
   - 内排序
     - 插入类排序
       - 直接插入排序法
       - 希尔排序法
     - 选择类排序
       - 直接选择排序法
       - 堆排序法
     - ★交换类排序
       - 冒泡排序法
       - 快速排序法
     - 归并类排序
   - 外排序
3. 注意：
   - 稳定性.
   - 时间复杂度.

4> 趟数及每一趟的结果

插入类排序

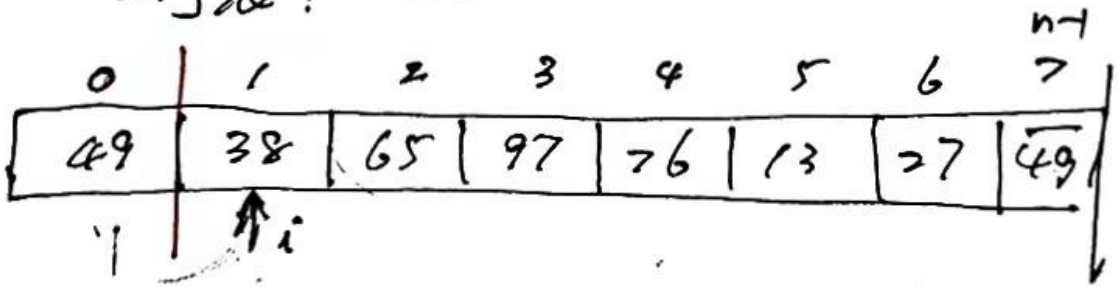
将一个关键字插入到一个有序表中的过程

1. 直接插入排序法：每次将无序表中的一个关键字插入到有序表中。

过程：49 38 65 97 76 13 27 $\overline{49}$

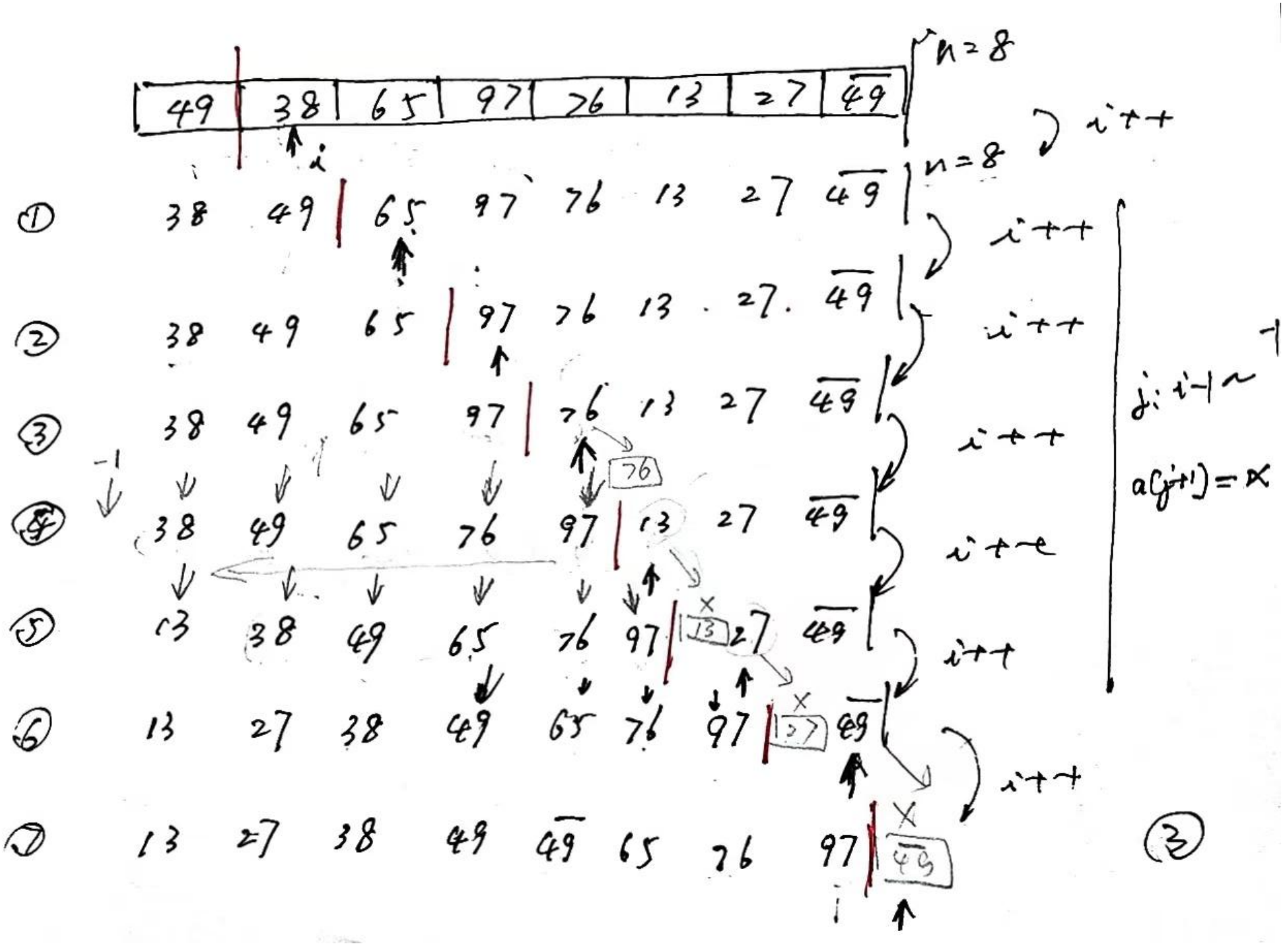
初始：有序表：[49]

无序表：[38. 65. 97. 76. 13 27 $\overline{49}$]



i：指向无序表的第一个元素，i：1 ~ n-1

有序表：0 ~ i-1.　　无序表：i —— n-1


n=8
49 38 65 97 76 13 27 49
i
i++
①
②
③
④
⑤
⑥
⑦
j: i-1 ~ 0
a[j+1]=x
-1
76
13
27
49
③

总结：(1) 共进行 $(n-1)$ 趟比较

(2) 时间复杂度：$O(n^2)$ —— 最好情况：可下降：$O(n)$

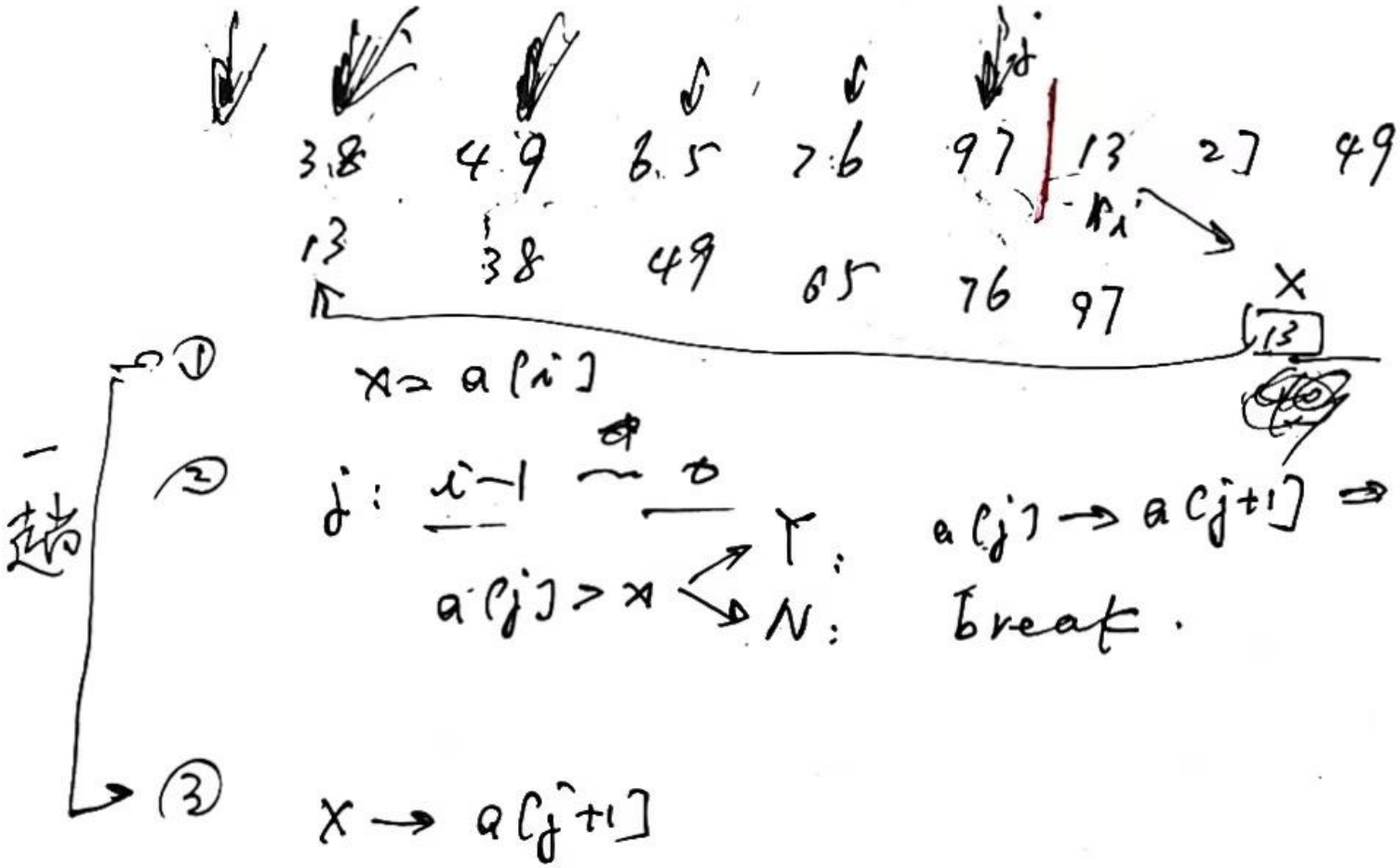
空间复杂度：$O(1)$

(3) 每趟比较次数 { 最好：1次
最好：$n-1$ 次

—— 因为采用的是顺序查找法

—— 可采用二分查找法提高效率

稳定性：好


38 49 65 76 97 13 27 49
13 38 49 65 76 97
X
13
x = a[i]
进
①
②
③
j: i-1 ~ 0
a[j] > x
Y: a[j] → a[j+1] ⇒ j--
N: break.
x → a[j+1]

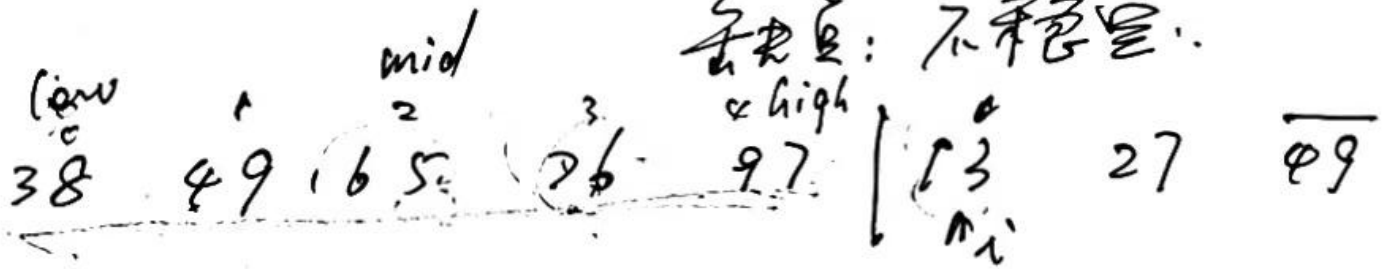
代码实现：

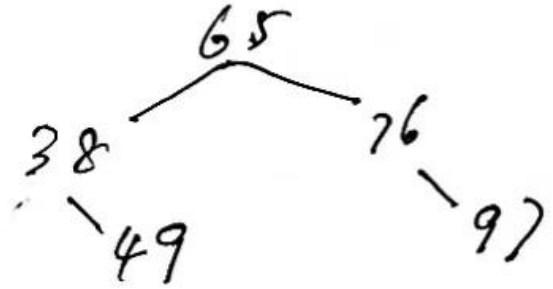
```
void InsertSort(int a[], int n)
{
    for(int i=1; i<n; ++i)
    {
        int x = a[i];
        int j;
        for(j=i-1; j>-1; --j)
        {
            if(a[j] > x) a[j+1] = a[j];
            else break;
        }
        a[j+1] = x;
    }
}
```

改进：{ 无序表：×
有序表：✓ —— 查找插入位置的过程

折半插入的排序 {
(1) 做折半插入的排序准备
(2) 在查找插入位置时采用折半查找，一提高查找效率。

缺点：不稳定。





low: 0
$mid = \frac{0+4}{2} = 2$
high: i-1
左：low=low, high=mid-1
右：low=mid+1, high=high

mid
low high

x=13

low mid high
0 2 4
38 49 65 76 97
0 1 2 3 4

① $mid = \frac{low+high}{2} = \frac{0+4}{2} = 2$

$a[mid] > x$ (65 > 13) $\Rightarrow$ 左区间：low = low, high = mid − 1

② $mid = \frac{low+high}{2} = \frac{0+1}{2} = 0$.

$a[mid] > x$ (38 > 13) $\rightarrow$ 左区间：low = low = 0, high = mid − 1 = −1.

low <= high 不成立 $\Rightarrow$ a[high+1] = x

0 −1

8

$x = 42$

	low	mid	high	mid			high
	0	1	2	3	4	5	6
	13	27	38	49	65	76	97
					49	65, 76	97

low = 0

high = i − 1

① $mid = \frac{low+high}{2} = \frac{0+6}{2} = 3$

$\underset{49}{a[mid]} > \underset{42}{x} \longrightarrow$ 左区 $\begin{cases} low = low \\ high = mid - 1 \end{cases}$

② $mid = \frac{low+high}{2} = \frac{0+2}{2} = 1$

$\underset{27}{a[mid]} < \underset{42}{x} \Rightarrow$ 右区 $\begin{cases} low = mid + 1 = 2 \\ high = high = 2 \end{cases}$

③ $mid = \frac{low+high}{2} = 2 \Rightarrow$ 右区 $\begin{cases} low = mid + 1 = 3 \\ high = high = 2 \end{cases}$

$\underset{38}{a[mid]} < \underset{42}{x}$

代码：

```
for(int i=1; i<n; ++i){
    x=a[i];
    //查找插入位置：
    low=0; high=i-1;
    while(low<=high){
        int mid=(low+high)/2;
        if(a[mid]<x)  low=mid+1;
        else  high=mid-1;
    }
    //移动元素：
    for(j=i-1; j>high; --j){
        a[j+1]=a[j];
    }
    a[high+1]=x;
}
```

时间复杂度：$O(n^2)$

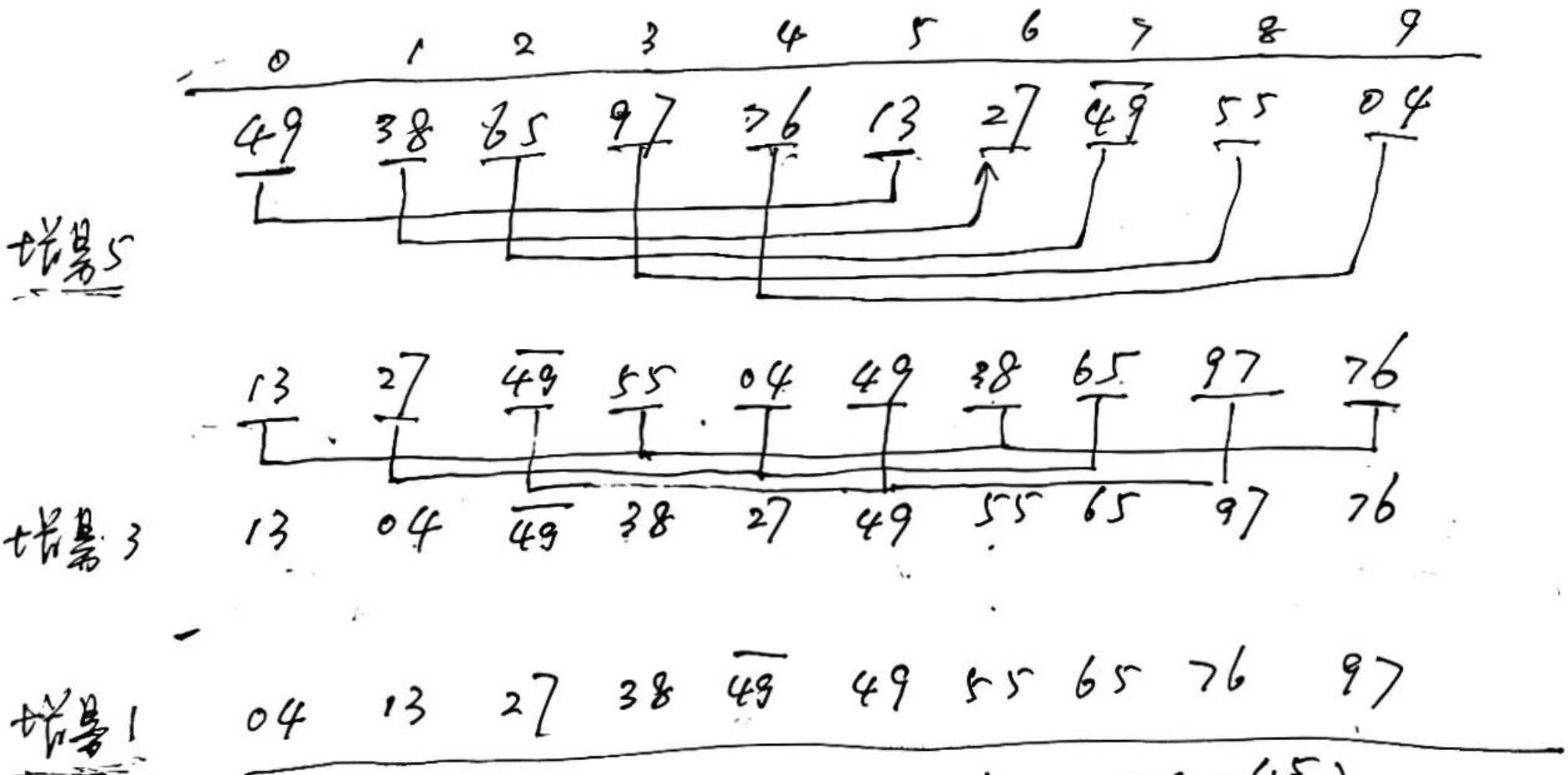
再改进：　　希尔排序法（缩小增量排序）

分析：若n值较小，直接插入排序法的算法简单使得排序效率较高。但随着n值呈几何级增大时，效率会呈几何级下降。（通过增量值完成）

改进：将待排序列划分成若干个子序列，每个子序列进行直接插入排序。

⇒直到整个序列基本有序

⇒最后再进行一次插入排序。

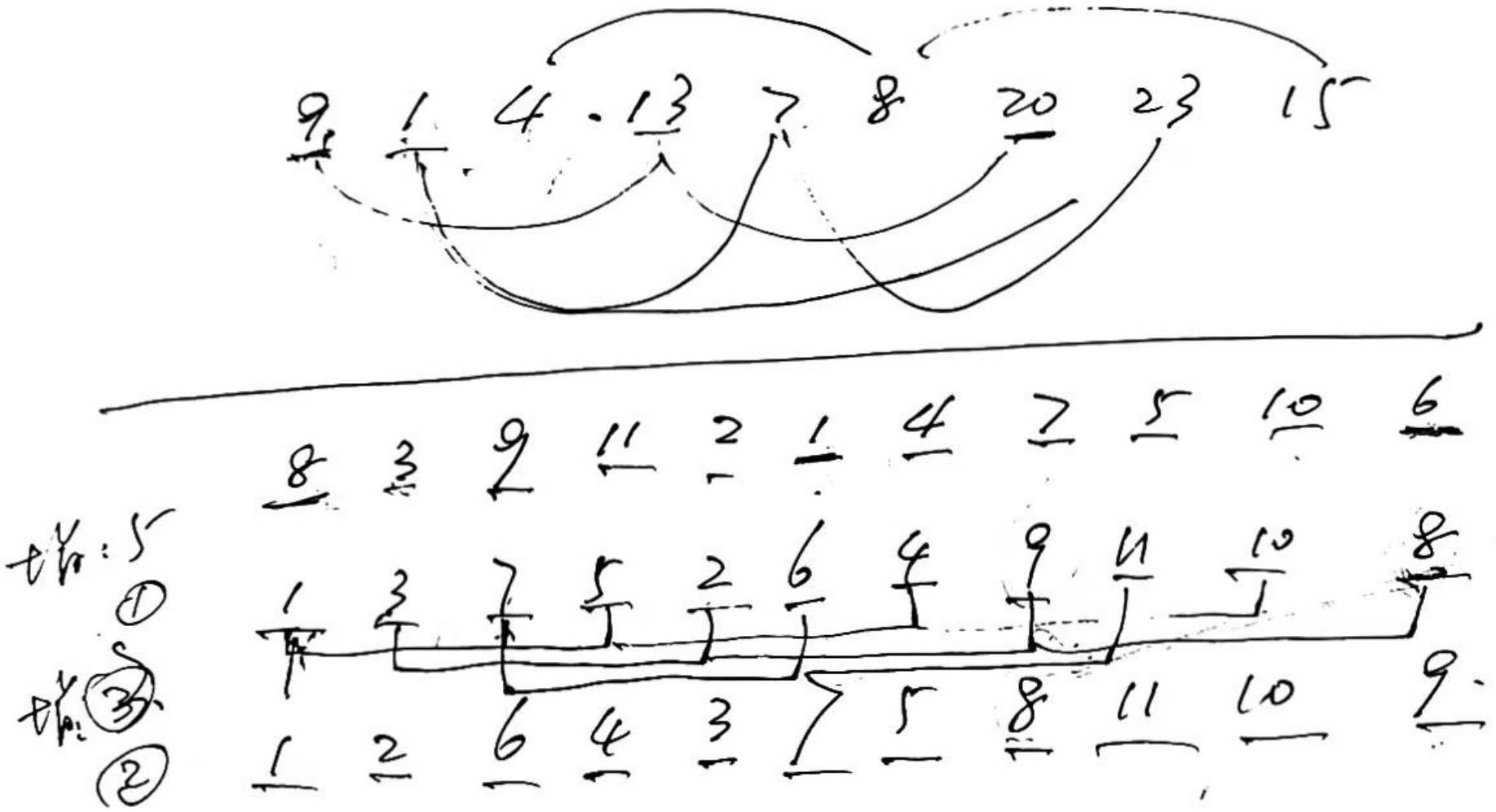


(最后再做一次直插的)

稳定性: NO. 时间: $O(n^{1.5})$.

☆ 希尔排序对目前为止尚未找到一个好的增量序列

☆ 最后增量必须是1.


9 1 4 13 7 8 20 23 15
8 3 9 11 2 1 4 7 5 10 6
增量：5
① 1 3 7 5 2 6 4 9 11 10 8
增量：3
② 1 2 6 4 3 5 8 11 10 9

5.3