

第八章 排序技术

8-1 排序概述

讲什么？



排序的定义



排序算法的性能



排序的类定义

排序的定义

数据元素、结点、顶点



📌 **排序**: 给定一组**记录**的集合 $\{r_1, r_2, \dots, r_n\}$, 其相应的关键码分别为 $\{k_1, k_2, \dots, k_n\}$, 将这些记录排列为 $\{r_{s1}, r_{s2}, \dots, r_{sn}\}$ 的**序列**, 使得相应的关键码满足 $k_{s1} \leq k_{s2} \leq \dots \leq k_{sn}$ (或 $k_{s1} \geq k_{s2} \geq \dots \geq k_{sn}$)

升序 (非降序)

降序 (非升序)

📌 **排序码**: 排序的依据
简单起见, 也称关键码

🕒 **排序的数据模型是什么?**
排序是对**线性结构**的一种操作

职工号	姓名	性别	年龄	工作时间
0001	王刚	男	48	1990.4
0002	张亮	男	35	2003.7
0003	刘楠	女	57	1979.9
0004	齐梅	女	35	2003.7
0005	李爽	女	56	1982.9

正序、逆序

✚ 正序：待排序序列中的记录已按关键码排好序



✚ 逆序（反序）：待排序序列中记录的顺序与排好序的顺序相反



✚ 趟：在排序过程中，将待排序的记录序列扫描一遍称为一趟

算法的稳定性

✚ **排序算法的稳定性**：假定在待排序的记录序列中存在多个具有相同关键码的记录，若经过排序，这些记录的**相对次序**保持不变，则称这种排序算法**稳定**，否则称为**不稳定**

学号	姓名	高数	英语	语文
0001	王 军	85	68	88
0002	李 明	64	72	92
0003	汤晓影	85	78	86
...



学号	姓名	高数	英语	语文
0001	李 明	64	68	88
0002	王 军	85	72	92
0003	汤晓影	85	78	86
...

排序算法的稳定性只是算法的一种属性，且由具体算法决定

排序的分类

✚ 根据排序过程中所有记录是否全部放在内存中，排序方法分为：

- (1) 内排序：在排序的整个过程中，待排序的所有记录全部放在内存中
- (2) 外排序：待排序的记录个数较多，整个排序过程需要在内外存之间多次交换数据才能得到排序的结果

✚ 根据排序方法是否建立在关键码比较的基础上，排序方法分为：

(1) 基于比较：主要通过关键码之间的比较和记录的移动实现

- | | | | |
|----------|---------|----------|-------------|
| ① 插入排序； | ② 交换排序； | ③ 选择排序； | ④ 归并排序 |
| └ 直接插入排序 | └ 起泡排序 | └ 简单选择排序 | └ 二路归并递归算法 |
| └ 希尔排序 | └ 快速排序 | └ 堆排序 | └ 二路归并非递归算法 |

(2) 不基于比较：根据待排序数据的特点所采取的其他方法

排序算法的性能



如何衡量排序算法的性能呢？

(1) 时间性能：排序算法在**各种情况（最好、最坏、平均）**下的时间复杂度。

基于比较的内排序在排序过程中的基本操作：

① **比较**：关键码之间的比较；

② **移动**：记录从一个位置移动到另一个位置。

(2) 空间性能：排序过程中占用的辅助存储空间。

辅助存储空间是除了存放待排序记录占用的存储空间之外，执行算法所需要的其他存储空间。

排序的类定义

```
class Sort
{
public:
    Sort(int r[ ], int n);
    ~Sort( );
    void InsertSort( );
    void ShellSort( );
    void BubbleSort( );
    void QuickSort(int first, int last);
    void SelectSort( );
    void HeapSort( );
    void MergeSort1(int first, int last);
    void MergeSort2( );
    void Print( );
```

📍 不失一般性，做如下约定：

- (1) 进行升序排序
- (2) 记录只有排序码一个数据项
- (3) 采用顺序存储

```
private:
    int Partition(int first, int last);
    void Sift(int k, int last);
    void Merge(int first1, int last1, int last2);
    void MergePass(int h);

    int *data;
    int length;
};
```


插入类排序

8-2-1 直接插入排序

讲什么？



直接插入排序的基本思想



直接插入排序的算法



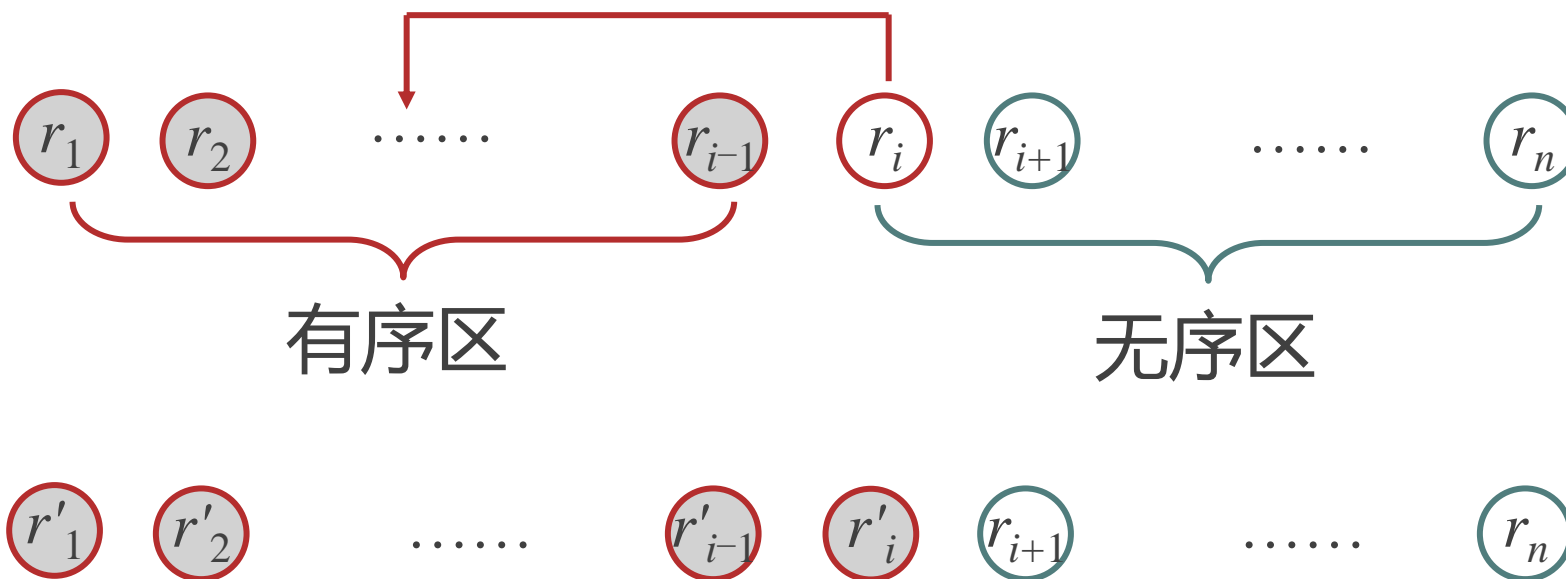
直接插入排序的时空性能



直接插入排序的稳定性

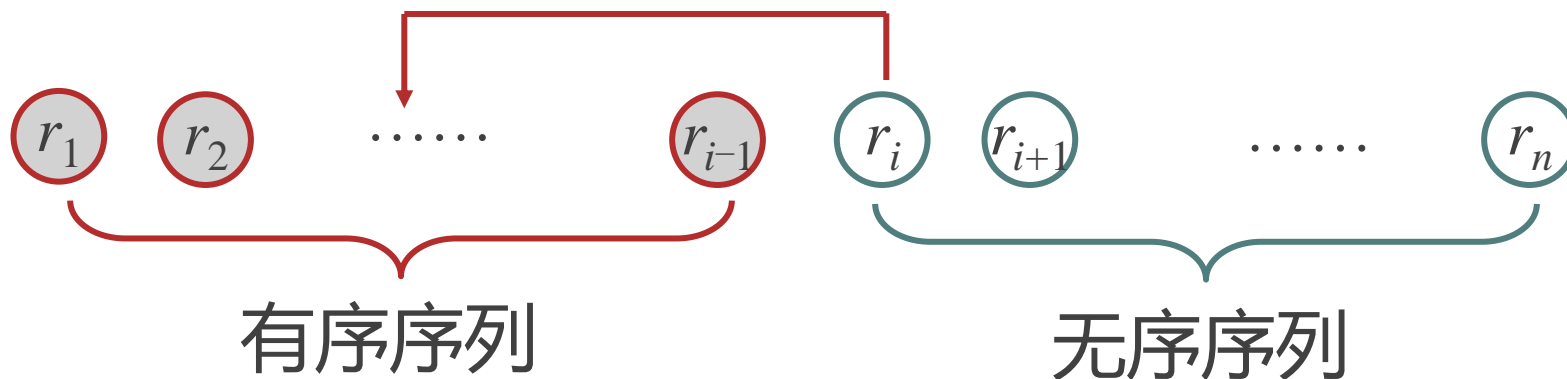
基本思想

📖 直接插入排序的基本思想：**依次**将待排序序列中的每一个记录插入到**已排好序**的序列中，直到全部记录都排好序。



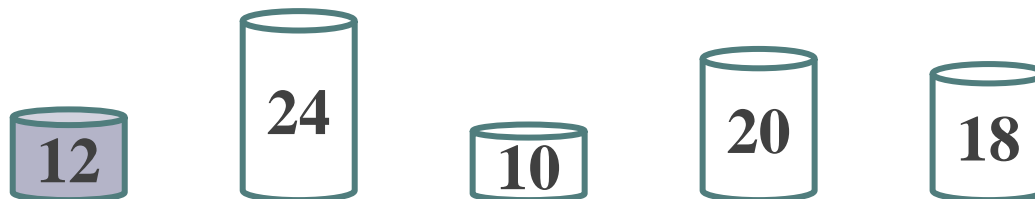
在插入第 i ($i > 1$) 个记录时，前面的 $i-1$ 个记录已经排好序

关键问题



直接插入排序在插入第 i 个记录时，前面的 $i-1$ 个记录已经排好序

待排序序列




如何构造初始的有序序列？

算法描述

待排序序列



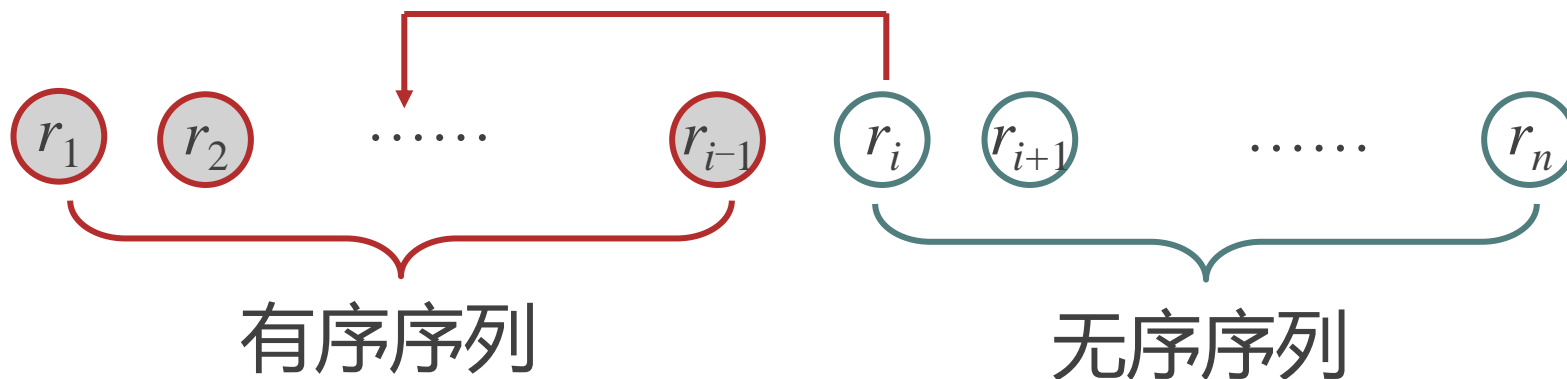
 **解决方法：**将第 1 个记录看成是初始有序序列，
然后从第 2 个记录起依次插入到有序序列中，直至将第 n 个记录插入。

 **算法描述：**

```
for (i = 1; i < length; i++)  
{  
    插入data[i];           //第 i 趟直接插入排序;  
}
```

 如何构造初始的有序序列？

关键问题



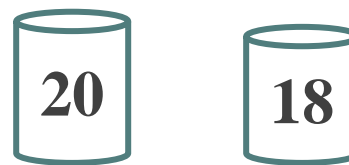
直接插入排序在插入第 i 个记录时，前面的 $i-1$ 个记录已经排好序



如何将第 i 个记录插入到有序序列中的合适位置？

算法描述

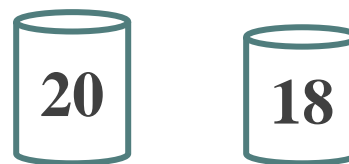
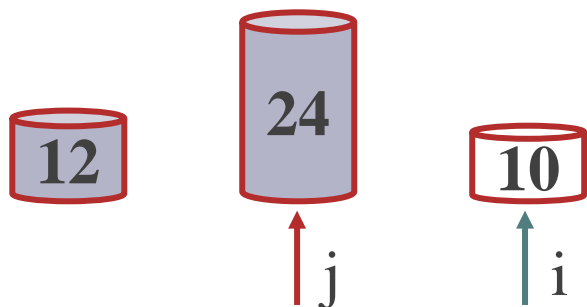
待排序序列



暂存单元

temp

第一趟排序结果



第二趟排序结果



算法描述:

```
temp = data[i]; j = i - 1;  
while (temp < data[j])  
{  
    data[j+1] = data[j]; j--;  
}  
data[j + 1] = temp;
```



查找下标初始化为多少?



循环条件是什么?



退出循环, 记录data[i]的最终位置在哪里?

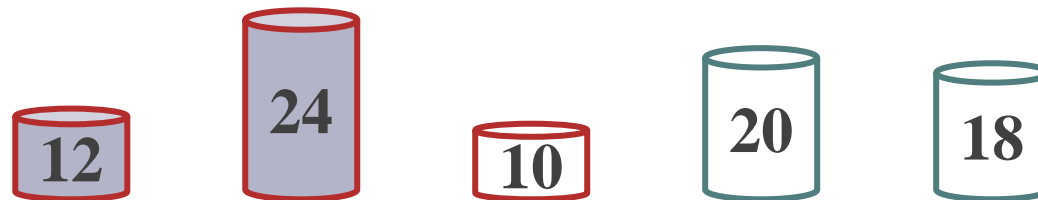
运行实例

<https://visualgo.net/en/sorting>

待排序序列



第一趟排序结果



第二趟排序结果



第三趟排序结果



第四趟排序结果



算法描述

```
void Sort :: InsertSort( )
{
    int i, j, temp;
    for (i = 1; i < length; i++)
    {
        temp = data[i];
        j = i - 1;
        while (j >= 0 && temp < data[j])
        {
            data[j + 1] = data[j];
            j--;
        }
        data[j + 1] = temp;
    }
}
```

可换作哨兵data[0]

比较次数



比较语句？ 执行次数？



最好情况： $n-1$ 次



最坏情况： $1+2+3+\dots+n-1$ 次



算法描述

```
void Sort :: InsertSort( )
{
    int i, j, temp;
    for (i = 1; i < length; i++)
    {
        temp = data[i];
        j = i - 1;
        while (j >= 0 && temp < data[j])
        {
            data[j + 1] = data[j];
            j--;
        }
        data[j + 1] = temp;
    }
}
```

可换作哨兵data[0]

移动次数



移动语句？ 执行次数？



最好情况：0次



最坏情况：(1+1)+(2+1)+...+(n-1+1)次



时间性能

📜 最好情况：正序 $O(n)$

📎 比较次数： $n-1$ 次

📎 移动次数：0次

📜 最坏情况：逆序 $O(n^2)$

📎 比较次数： $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$ 次

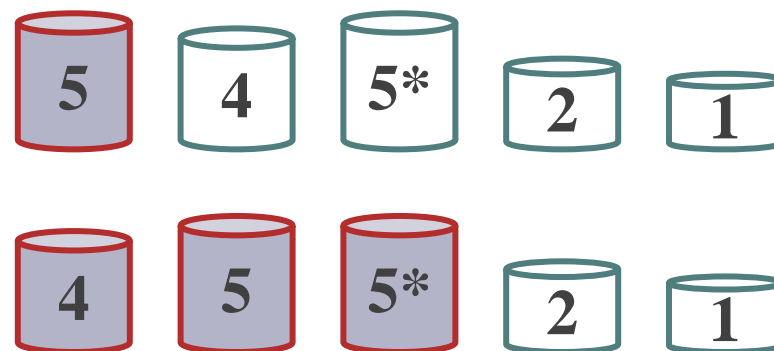
📎 移动次数： $\sum_{i=1}^{n-1} (i+1) = \frac{(n+2)(n-1)}{2}$ 次

📜 平均情况：随机排列， $O(n^2)$

空间性能

📜 空间性能： $O(1)$

📜 稳定性：稳定

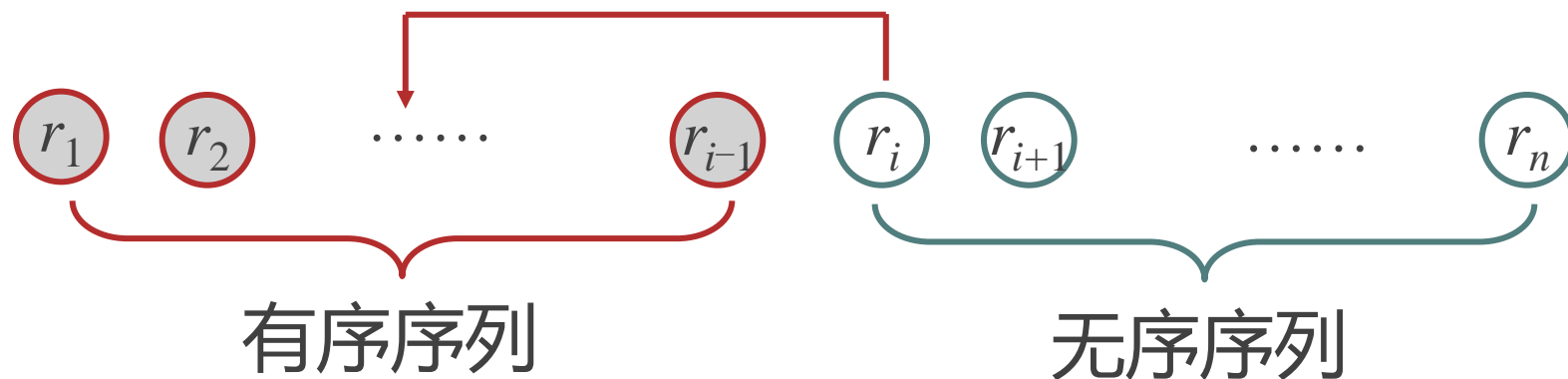


```
while (j >= 0 && temp < data[j])  
{  
    data[j + 1] = data[j];  
    j--;  
}
```

插入类排序

8-2-2 折半插入排序

折半插入排序（自学）



因为左边是有序，所以可以使用折半查找在有序区域很快找到插入位置

折半插入排序虽然可以减少关键码的比较次数，但是并不减少排序元素的移动次数。
比较次数为 $O(n \log n)$ ，移动次数为 $O(n^2)$

插入类排序

8-2-3 希尔排序

讲什么？



希尔排序改进的着眼点



希尔排序的基本思想



希尔排序的算法



希尔排序的时空性能



希尔排序的稳定性

希尔排序

相较于插入排序，1次比较最多只能移动1个元素，希尔排序可以对位置相隔较大距离的元素进行比较，使得元素在比较后能够一次跨过较大的距离。

基本思想：对待排记录序列先作“宏观”调整，再作“微观”调整。先将待排序记录划分为若干个子序列，并对这些子序列进行直接插入排序，待整个序列接近有序时，再对其进行直接插入排序。

这样做充分利用了插入排序在表的长度较小、表接近有序时性能优越的特点。



希尔排序

排序步骤：

- 先给定一组严格递减的正整数增量 d_0, d_1, \dots, d_{t-1} ，且取 $d_{t-1} = 1$ 。
- 对于 $i=0, 1, \dots, t-1$ ，进行下面各遍的处理：
 - ◆ 将序列分成 d_i 组，每组中结点的下标相差 d_i
 - ◆ 对每组节点使用插入排序

希尔排序（shell sort）是一种分组插入排序方法，也称为缩小增量排序。

希尔排序过程

对关键字序列{16,25,12,30,47,11,16,36,9,18,31}进行升序排列，设置增量序列为 $d = \{5, 3, 1\}$

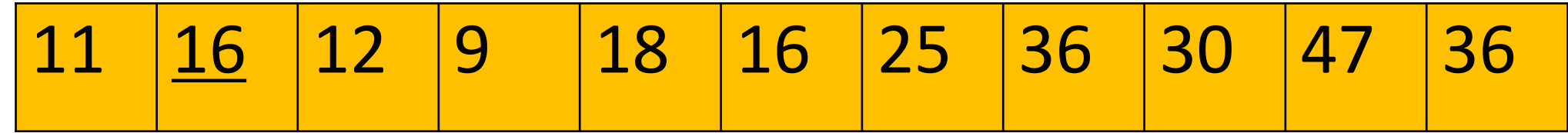
第一趟希尔排序，设增量 $d = 5$

0	1	2	3	4	5	6	7	8	9	10
16	25	12	30	47	11	<u>16</u>	36	9	18	31

11	<u>16</u>	12	9	18	16	25	36	30	47	31
----	-----------	----	---	----	----	----	----	----	----	----

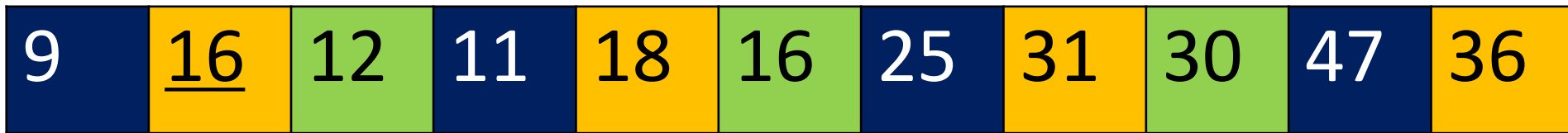


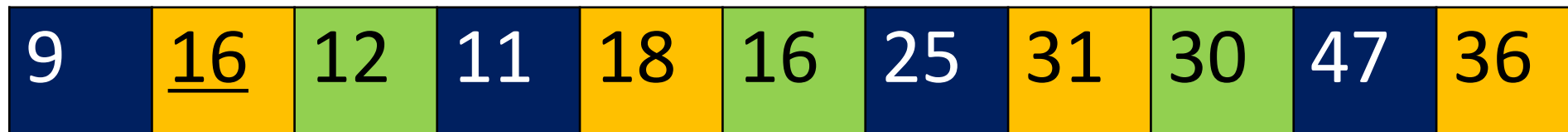
去掉分组
标记



第二趟希尔排序，设增量 $d = 3$

0 1 2 3 4 5 6 7 8 9 10





去掉分组
标记



第三趟希尔排序，设增量 $d = 1$



原始:



观察16和16的相对位置，可发现是希尔排序是不稳定的

算法描述

```
void sort::shellsort(){
    for (d=length/2;d>=1;d=d/2){
        //d不同，性能会有变化，此处d用的是折半策略
        for (i=d;i<length;i++){
            temp = data[i];
            for (j=i-d;j >= 0 && temp < data[j]; j = j - d)
                data[j + d] = data[j];

            data[j + d] = temp;
        }
    }
}
```



查找下标初始化为多少？循环条件是什么？



退出循环，记录data[i]的最终位置在哪里？

时空性能



时间性能：依赖于增量序列

d增量序列	最坏情况下复杂度
$n / 2^i$	(n^2)
$2^k - 1$	$(n^{3/2})$
$2^i 3^i$	$(n \log^2 n)$

研究表明，希尔排序的时间性能在 $O(n^2)$ 和 $O(n \log_2 n)$ 之间



空间性能： $O(1)$ ——暂存单元



稳定性：不稳定

1. 如果某种排序算法是不稳定的，则该排序方法没有实际应用价值。

☐ A 正确

☒ B 错误

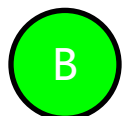
提交

2. 排序算法只能在顺序存储结构上进行，链接存储无法对记录进行排序。



A

正确



B

错误

提交

3. 所有排序算法在排序过程中的基本操作都是比较和移动。

☐ A 正确

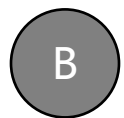
☒ B 错误

提交

4. 排序码是排序的依据，排序码通常是关键码。



正确



错误

提交

1. 直接插入排序的主要操作是将元素插入到某个子序列中。

☐ A 正确

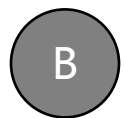
☒ B 错误

提交

2. 无论待排序序列的初始状态如何，直接插入排序都会执行 $n-1$ 趟。



正确



错误

提交

3. 直接插入排序在最好情况下需要较少的比较次数和移动次数, 时间复杂度是 $O(n^2)$ 。

☐ A 正确

☒ B 错误

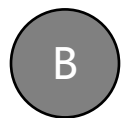
提交

4. 对于直接插入排序，最后一趟有可能改变每个元素的存储位置。



A

正确



B

错误

提交

5. 对于待排序记录序列{12, 25, 18, 15, 10}, 给出直接插入每一趟的结果。

1. 希尔排序将待排序序列逐段分割成若干个子序列，在子序列内部分别进行直接插入排序。

☐ A 正确

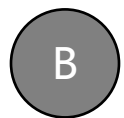
☒ B 错误

提交

2. 在希尔排序过程中，每一趟待排序的子序列长度逐渐增大。



正确



错误

提交

3. 希尔排序最后一趟的增量是（ ）。

A

1

B

2

C

3

D

不一定

提交

4. 希尔排序本质上属于插入排序，插入最后一个记录时，可能会改变所有元素的存储位置。

☐ A 正确

☒ B 错误

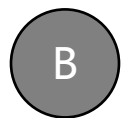
提交

5. 希尔排序是不稳定的排序方法。



A

正确



B

错误

提交

6. 对于待排序记录序列{25, 18, 30, 20, 15, 12, 18, 45, 35, 10}, 给定增量为4、2、1, 写出希尔排序每一趟的结果。