数据结构与算法（八）

张铭 主讲

采用教材：张铭，王腾蛟，赵海燕 编写
高等教育出版社，2008. 6（“十一五”国家级规划教材）

http://www.jpk.pku.edu.cn/pkujpk/course/sjjg

大 纲

- 8.1 排序问题的基本概念
- **8.2 插入排序 (Shell 排序)**
- 8.3 选择排序 (堆排序)
- 8.4 交换排序
  - 8.4.1 冒泡排序
  - 8.4.2 快速排序
- 8.5 归并排序
- 8.6 分配排序和索引排序
- 8.7 排序算法的时间代价
- 内排序知识点总结

8.2 插入排序

- 8.2.1 直接插入排序
- 8.2.2 Shell 排序

插入排序动画

插入排序算法

12	34	45	78	34′

```
template <class Record>
void ImprovedInsertSort (Record Array[], int n){
//Array[] 为待排序数组，n 为数组长度
    Record TempRecord;                  // 临时变量
    for (int i=1; i<n; i++){            // 依次插入第 i 个记录
        TempRecord = Array[i];
        //从 i 开始往前寻找记录 i 的正确位置
        int j = i-1;
        //将那些大于等于记录 i 的记录后移
        while ((j>=0) && (TempRecord < Array[j])){
            Array[j+1] = Array[j];
            j = j - 1;
        }
        //此时 j 后面就是记录 i 的正确位置，回填
        Array[j+1] = TempRecord;
    }
}
```

算法分析

- 稳定
- 空间代价：Θ(1)
- 时间代价：
  - 最佳情况：n-1 次比较，2(n-1) 次移动，Θ(*n*)
  - 最差情况： Θ(n^2)
    - 比较次数为 $\sum_{i=1}^{n-1} i = n(n-1)/2 = \Theta(n^2)$
    - 移动次数为 $\sum_{i=1}^{n-1} (i+2) = (n-1)(n+4)/2 = \Theta(n^2)$
  - 平均情况：Θ(n^2)

8.2.2 Shell排序

- 直接插入排序的两个性质：
  - 在最好情况（序列本身已是有序的）下时间代价为 $\Theta(n)$
  - 对于短序列，直接插入排序比较有效
- Shell 排序有效地利用了直接插入排序的这两个性质

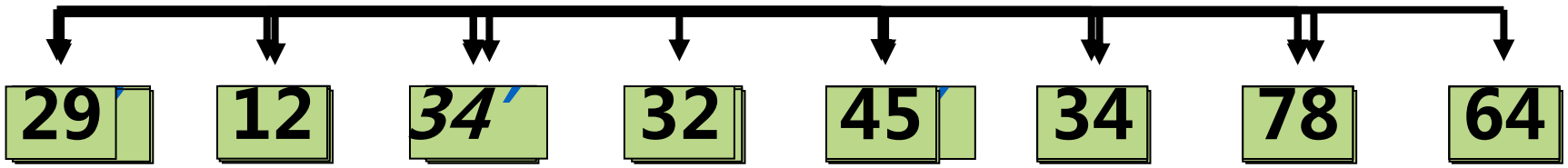
8.2.2 Shell排序

Shell排序算法思想

- 先将序列转化为若干小序列，在这些小序列内进行插入排序
- 逐渐增加小序列的规模，而减少小序列个数，使得待排序序列逐渐处于更有序的状态
- 最后对整个序列进行扫尾直接插入排序，从而完成排序

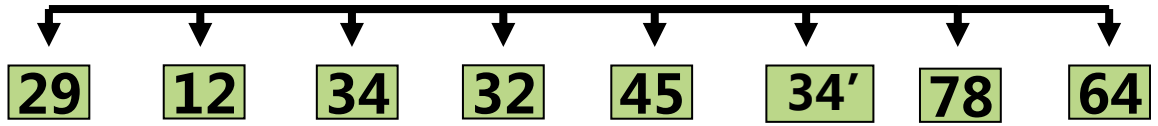
8.2.2 Shell排序

Shell排序动画



“增量每次除以2递减”的Shell 排序

```
template <class Record>
void ShellSort(Record Array[], int n) {
// Shell排序，Array[]为待排序数组，n为数组长度
    int i, delta;
    // 增量delta每次除以2递减
    for (delta = n/2; delta>0; delta /= 2)
        for (i = 0; i < delta; i++)
            // 分别对delta个子序列进行插入排序
            //“&”传 Array[i]的地址，数组总长度为n-i
            ModInsSort(&Array[i], n-i, delta);
    // 如果增量序列不能保证最后一个delta间距为1
    // 可以安排下面这个扫尾性质的插入排序
    // ModInsSort(Array, n, 1);
}
```



8.2.2 Shell排序

针对增量而修改的插入排序算法

```
template <class Record> // 参数delta表示当前的增量
void ModInsSort(Record Array[], int n, int delta) {
    int i, j;
    // 对子序列中第i个记录，寻找合适的插入位置
    for (i = delta; i < n; i += delta)
        // j以dealta为步长向前寻找逆置对进行调整
        for (j = i; j >= delta; j -= delta)  {
            if (Array[j] < Array[j-delta]) // 逆置对
                swap(Array, j, j-delta);// 交换
            else break;
        }
}
```

8.2.2 Shell排序

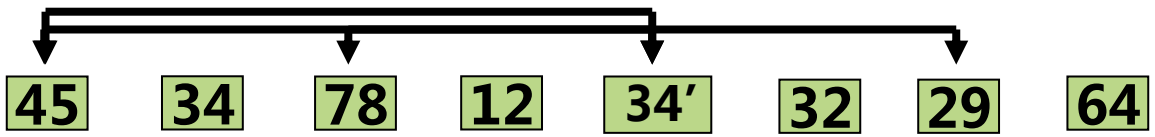
算法分析

- 不稳定
- 空间代价：Θ(1)
- 时间代价
  - 增量每次除以2递减，Θ(n^2)
- 选择适当的增量序列
  - 可以使得时间代价接近 Θ(n)

8.2.2 Shell排序

Shell 排序选择增量序列

- 增量每次除以2递减
  - 效率仍然为 $\Theta(n^2)$
- 问题：选取的增量之间并不互质
  - 间距为 2^{k-1} 的子序列，都是由那些间距为 2^k 的子序列组成的
  - 上一轮循环中这些子序列都已经排过序了，导致处理效率不高



Hibbard 增量序列

- Hibbard 增量序列
  - $\{2^k-1，2^{k-1}-1，…，7，3，1\}$
- Shell(3) 和 Hibbard 增量序列的 Shell 排序的效率可以达到 $\Theta(n^{3/2})$
- 选取其他增量序列还可以更进一步减少时间代价

8.2.2 Shell排序

Shell最好的代价

- 呈 2^p3^q 形式的一系列整数：
  - 1, 2, 3, 4, 6, 8, 9, 12
- $\Theta(n \log_2 n)$

8.2.2 Shell排序

思考

- 1. 插入排序的变种
  - 发现逆序对直接交换
  - 查找待插入位置时，采用二分法

- 2. Shell 排序中增量作用是什么？增量为2和增量为3的序列，哪个更好？为什么？

- 3. Shell 排序的每一轮子序列排序可以用其他方法吗？

数据结构与算法

谢谢聆听

国家精品课“数据结构与算法”
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

张铭，王腾蛟，赵海燕
高等教育出版社，2008. 6。“十一五”国家级规划教材