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

8.4 交换排序

- 8.4.1 冒泡排序
- 8.4.2 快速排序

8.4.1 冒泡排序

- 算法思想
  - 不停地比较相邻的记录，如果不满足排序要求，就交换相邻记录，直到所有的记录都已经排好序
- 检查每次冒泡过程中是否发生过交换，如果没有，则表明整个数组已经排好序了，排序结束
  - 避免不必要的比较

- 冒泡排序之舞
  http://v.youku.com/v_show/id_XMjU4MTg3MTU2.html

8.4.1 冒泡排序

冒泡排序动画

45	34	78	12	34′	32	29	64

冒泡排序算法

```
template <class Record>
void BubbleSort(Record Array[], int n) {
    bool NoSwap;                         // 是否发生了交换的标志
    int i, j;
    for (i = 0; i < n-1; i++) {
        NoSwap = true;                   // 标志初始为真
        for (j = n-1; j > i; j--){
            if (Array[j] < Array[j-1]) { // 判断是否逆置
                swap(Array, j, j-1);     // 交换逆置对
                NoSwap = false           // 发生了交换，标志变为假
            }
        if (NoSwap)                      // 没发生交换，则已完成排好序
            return;
        }
    }
}
```

8.4.1 冒泡排序

算法分析

- 稳定
- 空间代价：Θ(1)
- 时间代价分析
  - 比较次数
    - 最少：$\Theta(n)$
    - 最多：

$$\sum_{i=1}^{n-1}(n-i)=n(n-1)/2=\Theta(n^2)$$

  交换次数最多为 $\Theta(n^2)$，最少为 0，平均为 $\Theta(n^2)$

- 时间代价结论
  - 最大，平均时间代价均为 $\Theta(n^2)$
  - 最小时间代价为 $\Theta(n)$：最佳情况下只运行第一轮循环

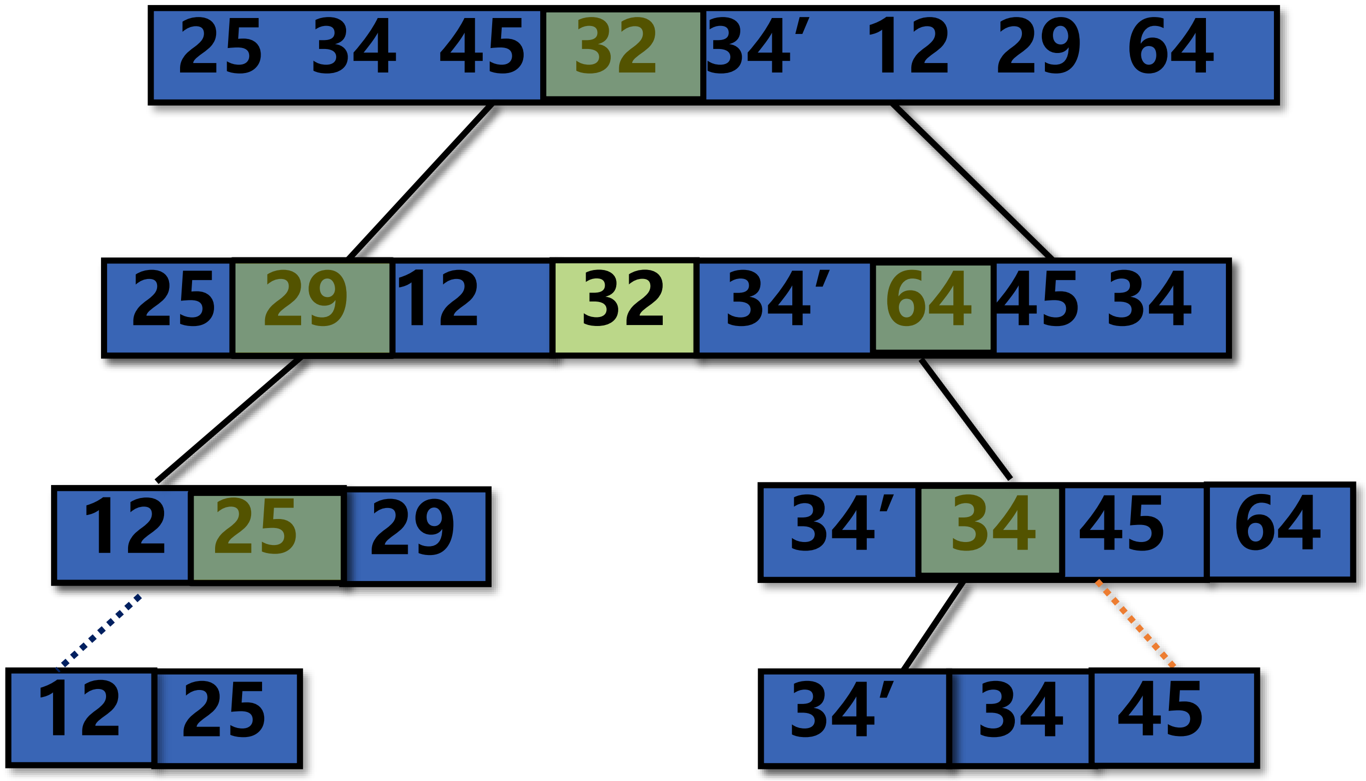
8.4.2 快速排序

- 算法思想
  - 选择轴值 (pivot)
  - 将序列划分为两个子序列 L 和 R，使得 L 中所有记录都小于或等于轴值，R 中记录都大于轴值
  - 对子序列 L 和 R 递归进行快速排序
- 20世纪十大算法
  - Top 10 Algorithms of the Century
  - 7. 1962 London 的 Elliot Brothers Ltd 的 Tony Hoare 提出的快速排序
- 基于分治法的排序：快速、归并

分治策略的基本思想

- 分治策略的实例
  - BST 查找、插入、删除算法
  - 快速排序、归并排序
  - 二分检索
- 主要思想
  - 划分
  - 求解子问题 (子问题不重叠)
  - 综合解

快速排序分治思想



最终排序结果： 12 25 29 32 34′ 34 45 64

轴值选择

- 尽可能使 L, R 长度相等
- 选择策略：
  - 选择最左边记录
  - 随机选择
  - 选择平均值

8.4.2 快速排序

一次分割过程

25	34	45	32	34’	12	29	64
i							j

- 选择轴值并存储轴值
- 最后一个元素放到轴值位置
- 初始化下标 i, j，分别指向头尾
- i 递增直到遇到比轴值大的元素，将此元素覆盖到 j 的位置；j 递减直到遇到比轴值小的元素，将此元素覆盖到 i 的位置
- 重复上一步直到 i==j，将轴值放到 i 的位置，完毕

快速排序算法

```
template <class Record>
void QuickSort(Record Array[], int left, int right)  {
// Array[]为待排序数组，left,right分别为数组两端
    if (right <= left)              // 只有0或1个记录，就不需排序
        return;
    int pivot = SelectPivot(left, right);         // 选择轴值
    swap(Array, pivot, right);                  // 轴值放到数组末端
    pivot = Partition(Array, left, right);        // 分割后轴值正确
    QuickSort(Array, left, pivot-1);            // 左子序列递归快排
    QuickSort(Array, pivot +1, right);            // 右子序列递归快排
}
int SelectPivot(int left, int right) {
   // 选择轴值，参数left,right分别表示序列的左右端下标
    return (left+right)/2;                        // 选中间记录作为轴值
}
```

分割函数

```
template <class Record>
int Partition(Record Array[], int left, int right) {
// 分割函数，分割后轴值已到达正确位置
    int l = left;                  // l 为左指针
    int r = right;                 // r 为右指针
    Record TempRecord = Array[r];    // 保存轴值
    while (l != r) {               // l, r 不断向中间移动，直到相遇
        // l 指针向右移动，直到找到一个大于轴值的记录
        while (Array[l] <= TempRecord && r > l)
            l++;
        if (l < r)  {              // 未相遇，将逆置元素换到右边空位
            Array[r] = Array[l];
            r--;                   //  r 指针向左移动一步
        }
```

8.4.2 快速排序

```
        // r 指针向左移动，直到找到一个小于轴值的记录
        while (Array[r] >= TempRecord && r > l)
            r--;
        if (l < r) {              // 未相遇，将逆置元素换到左空位
            Array[l] = Array[r];
            l++;                  // l 指针向右移动一步
        }
    } //end while
    Array[l] = TempRecord;  // 把轴值回填到分界位置 l 上
    return l;               // 返回分界位置l
}
```

时间代价

- 长度为n的序列，时间为 T(n)
  - T(0) = T(1) =1
- 选择轴值时间为常数
- 分割时间为 cn
  - 分割后长度分别为 i 和 n-i-1
  - 左右子序列 T(i) 和 T(n-i-1)
- 求解递推方程

$$T(n) = T(i) + T(n-1-i) + cn$$

最差情况

$$T(n) = T(n-1) + cn$$

$$T(n-1) = T(n-2) + c(n-1)$$

$$T(n-2) = T(n-3) + c(n-2)$$

$$\cdots$$

$$T(2) = T(1) + c(2)$$

- 总的时间代价为：

$$T(n) = T(1) + c\sum_{i=2}^{n} i = \Theta(n^2)$$

8.4.2 快速排序

最佳情况

$$T(n) = 2T(n/2) + cn$$

$$\frac{T(n)}{n} = \frac{T(n/2)}{n/2} + c$$

$$\frac{T(n/2)}{n/2} = \frac{T(n/4)}{n/4} + c$$

$$\frac{T(n/4)}{n/4} = \frac{T(n/8)}{n/8} + c$$

$$\cdots$$

$$\frac{T(2)}{2} = \frac{T(1)}{1} + c$$

$$\frac{T(n)}{n} = \frac{T(1)}{1} + c\log n$$

$$T(n) = cn\log n + n = \Theta(n\log n)$$

等概率情况

- 也就是说，轴值将数组分成长度为 0 和 n-1, 1 和 n-2，… 的子序列的概率是相等的，都为 1/n
- T(i) 和 T(n-1-i) 的平均值均为

$$T(i) = T(n-1-i) = \frac{1}{n}\sum_{k=0}^{n-1} T(k)$$

$$T(n) = cn + \frac{1}{n}\sum_{k=0}^{n-1}(T(k) + T(n-1-k)) = cn + \frac{2}{n}\sum_{k=0}^{n-1} T(k)$$

$$nT(n) = (n+1)T(n-1) + 2cn - c$$

$$T(n) = \Theta(n \log n)$$

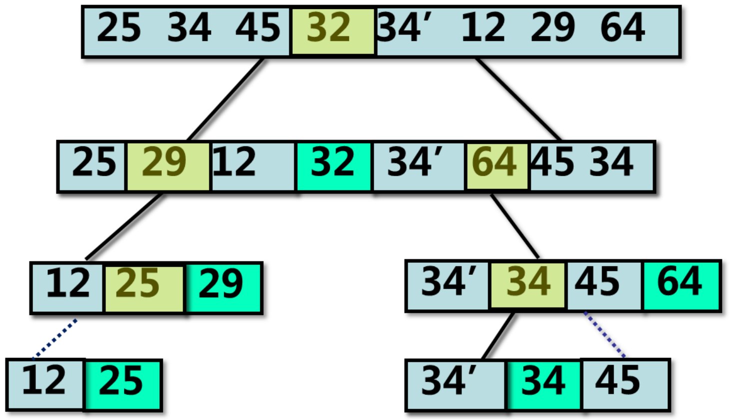
快速排序算法分析

- 最差情况：
  - 时间代价：$\Theta(n^2)$
  - 空间代价：$\Theta(n)$
- 最佳情况：
  - 时间代价：$\Theta(n\log n)$
  - 空间代价：$\Theta(\log n)$
- 平均情况：
  - 时间代价：$\Theta(n\log n)$
  - 空间代价：$\Theta(\log n)$

8.4.2 快速排序

思考

- 冒泡排序和直接选择排序哪个更优
- 快速排序为什么不稳定
- 快速排序可能的优化
  - 轴值选择 RQS
  - 小子串不递归 （阈值 28？ ）
  - 消除递归（用栈，队列？）



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