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）

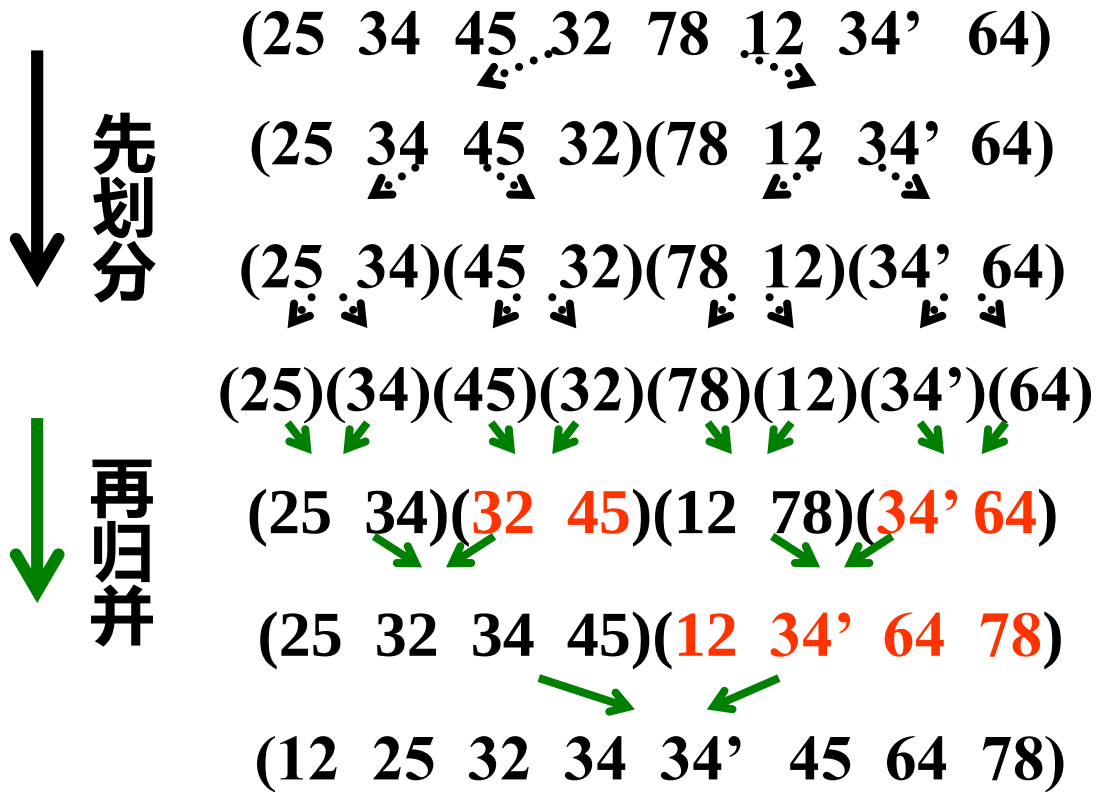
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg

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

归并排序思想

- 划分为两个子序列
- 分别对每个子序列归并排序
- 有序子序列合并



两路归并排序

```
template <class Record>
void MergeSort(Record Array[], Record TempArray[], int
left, int right) {
    // Array为待排序数组，left，right两端
    int middle;
    if (left < right) { // 序列中只有0或1个记录，不用排序
        middle = (left + right) / 2;   // 平分为两个子序列
        // 对左边一半进行递归
        MergeSort(Array,TempArray,left,middle);
        // 对右边一半进行递归
        MergeSort(Array, TempArray,middle+1,right);
        Merge(Array, TempArray,left,right,middle); // 归并
    }
}
```

归并函数

```
// 两个有序子序列都从左向右扫描，归并到新数组
template <class Record>
void Merge(Record Array[], Record TempArray[], int left,
int right, int middle) {
    int i, j, index1, index2;
    // 将数组暂存入临时数组
    for (j = left; j <= right; j++)
        TempArray[j] = Array[j];
    index1 = left;              // 左边子序列的起始位置
    index2 = middle+1;          // 右边子序列的起始位置
    i = left;                   // 从左开始归并
```

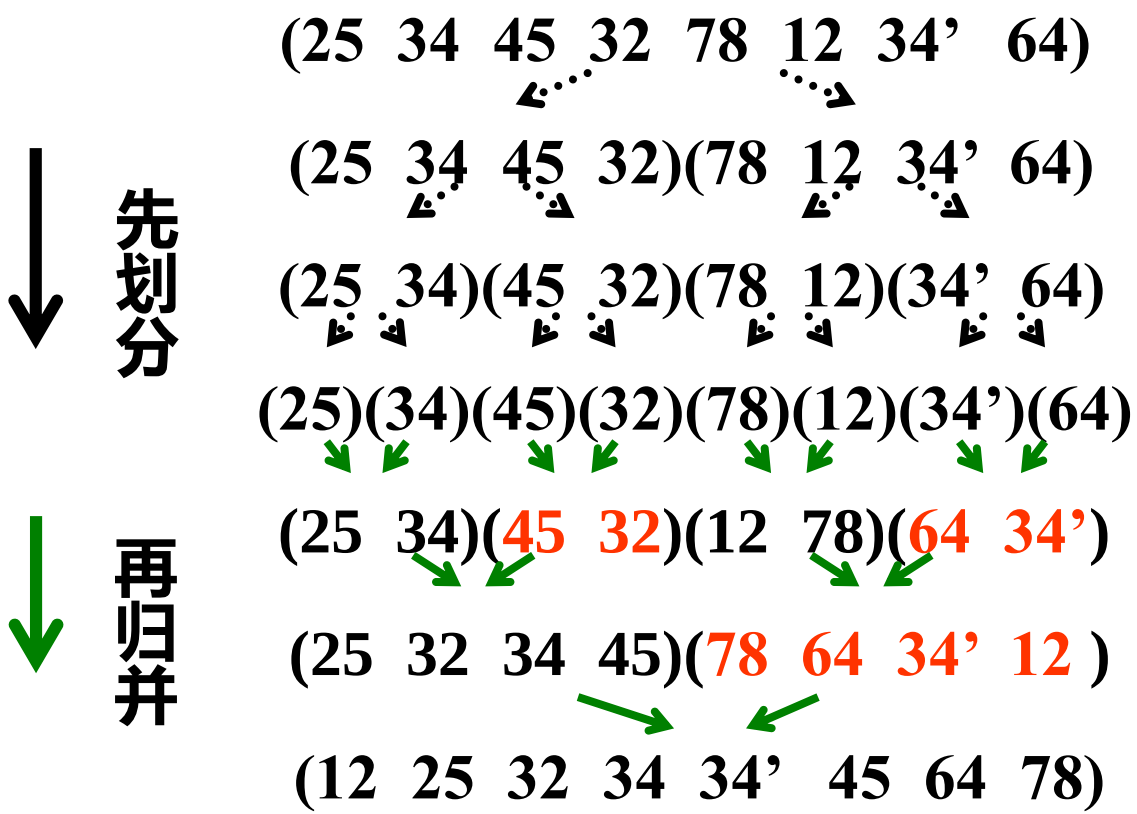
8.5 归并排序

```
while (index1 <= middle && index2 <= right)  {
    // 取较小者插入合并数组中
    if (TempArray[index1] <= TempArray[index2])
        Array[i++] = TempArray[index1++];
    else  Array[i++] = TempArray[index2++];
}
while (index1 <= middle)  // 只剩左序列，可以直接复制
    Array[i++] = TempArray[index1++];
while (index2 <= right)     // 与上个循环互斥，复制右序列
    Array[i++] = TempArray[index2++];
}
```

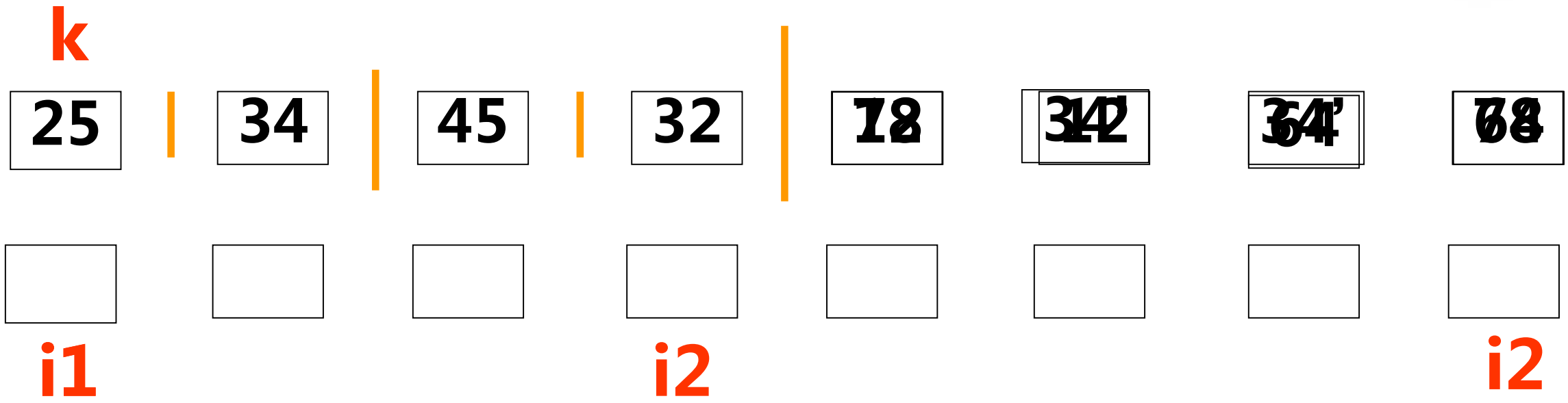
归并算法优化

- 同优化的快速排序一样，对基本已排序序列直接插入排序
- R.Sedgewick 优化：归并时从两端开始处理，向中间推进，简化边界判断

R.Sedgewick优化归并思想



8.5 归并排序



R.Sedgewick优化归并

优化的归并排序（阈值28）

```
template <class Record>
void ModMergeSort(Record Array[], Record TempArray[], int left, int
right) {      // Array为待排序数组，left，right两端
    int middle;
    if (right-left+1 > THRESHOLD) {   // 长序列递归
        middle = (left + right) / 2;        // 从中间划为两个子序列
        ModMergeSort(Array, TempArray ,left,middle);      // 左
        ModMergeSort(Array, TempArray ,middle+1,right);// 右
        // 对相邻的有序序列进行归并
        ModMerge(Array, TempArray ,left,right,middle);    // 归并
    }
    else  InsertSort(&Array[left],right-left+1);    // 小序列插入排序
}
```

优化的归并函数

```
template <class Record> void ModMerge(Record
Array[],Record TempArray[],int left,int right,int middle) {
    int index1,index2;                    // 两个子序列的起始位置
    int i,j,k ;
    for (i = left; i <= middle; i++)
        TempArray[i] = Array[i];          // 复制左边的子序列
    for (j = 1; j <= right-middle; j++) // 颠倒复制右序列
        TempArray[right-j+1] = Array[j+middle];
    for (index1=left, index2=right, k=left; k<=right;  k++)
        if (TempArray[index1] <= TempArray[index2])
            Array[k] = TempArray[index1++];
        else
            Array[k] = TempArray[index2--];
}
```

算法复杂度分析

- 空间代价：$\Theta(n)$
- 划分时间、排序时间、归并时间

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

- $T(1) = 1$
- 归并排序总时间代价也为
  - $\Theta(n\log n)$
- 不依赖于原始数组的输入情况，最大、最小以及平均时间代价均为 $\Theta(n\log n)$

思考

- 普通归并和 Sedgewick 算法都是稳定的吗？
- 两个归并算法哪个更优？
  - 二者的比较次数和赋值次数
  - 归并时子序列下标是否需要边界判断

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