专题　排序和查找

计算机学院

已经学过的排序算法（7种）

- 第2章
  - 计数排序
  - 选择排序
  - 冒泡排序
  - 插入排序（折半插入排序）
- 第5-6章
  - 箱子排序
  - 基数排序
- 第12章
  - 堆排序

已经学过的查找方法（5种）

- 第10章
  - 哈希查找
- 第14-15章
  - BST查找
  - AVL查找
  - 红黑树查找
  - B树查找

主要内容

- 归并排序
- 快速排序
- 希尔排序
- 排序算法的分析与比较
- 关于查找的讨论

归并排序

- 考虑用分治方法解决排序问题
  - 原子问题——n=1
  - 分解方法一
    - 前n-1个元素为集合A，最后一个为集合B
    - 对A递归地使用分治方法进行排序，B自然有序
    - A、B合并——

插入排序!

简单分治排序

- 分解方法二
  - 选出最大的元素作为B，剩余的作为A
  - 对A递归地进行排序
  - 无需合并——

选择排序！

- 性能差——划分不平衡

平衡划分——归并排序

- A——n/k个元素，B——n - n/k个元素
- k=2时，均匀划分
- A、B排序完成后，合并（merge）它们

归并排序

- 思想
  - 对于一个需要排序的数组A[0⋯n-1]，把它一分为二： A[0⋯n/2-1]和A[n/2⋯n-1]，并对每个子数组递归排序
  - 然后把这两个排好序的子数组合并为一个有序数组

算法描述

```
//Mergesort
if n>1
  copy A[0…n/2-1] to B[0…n/2-1]
  copy A[n/2…n-1] to C[0…n/2-1]
  Mergesort(B[0…n/2-1])
  Mergesort(C[0…n/2-1])
  Merge(B,C,A)
```

时间代价较小，空间消耗较多

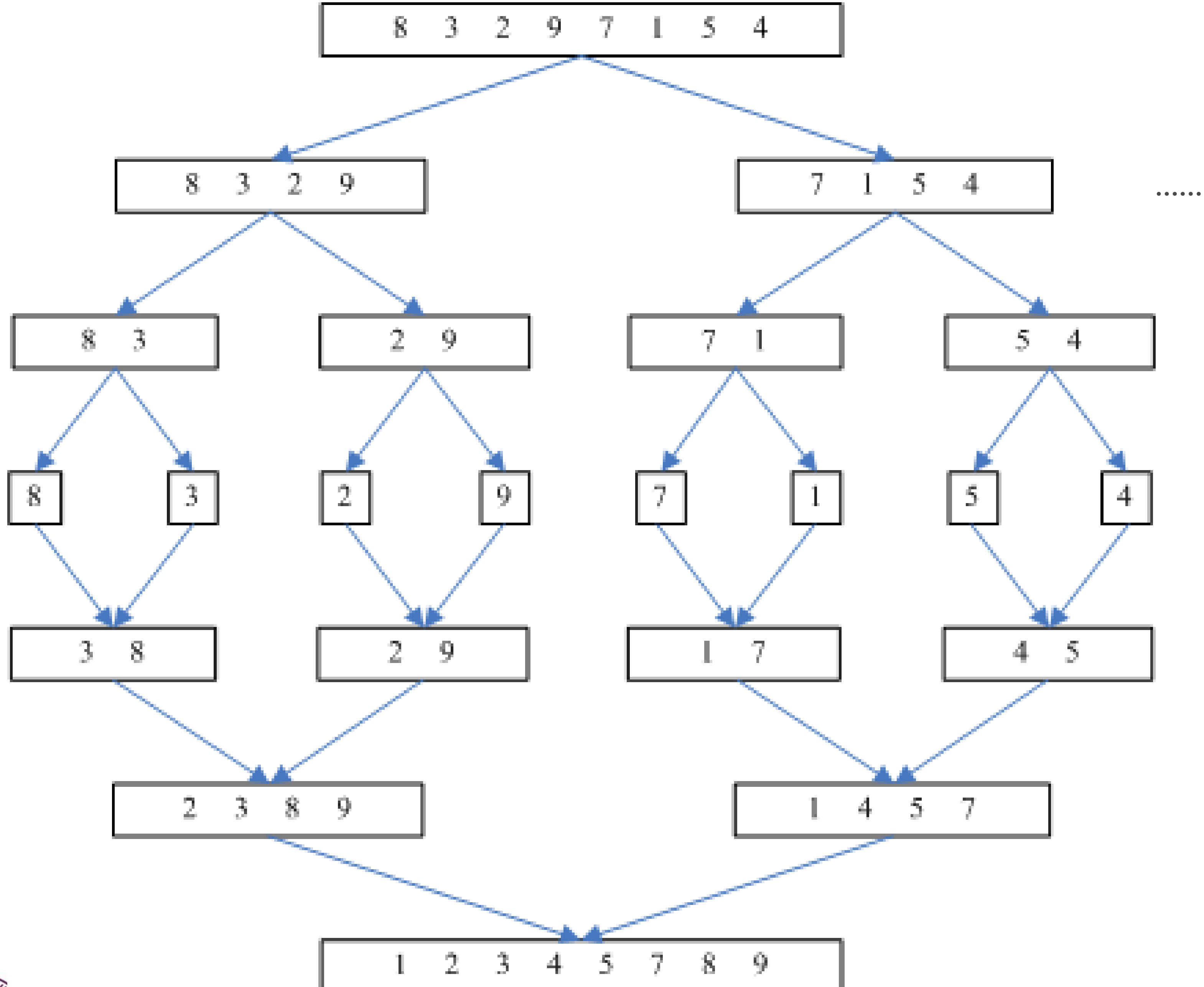
$$T(n) = O(n\log_2 n)$$

Merge算法

- 思想
  - 对两个有序数组的合并
  - 初始状态下，关注两个待合并数组的第一个元素
  - 然后比较这两个元素的大小，将较小的元素添加到一个新创建的数组中
  - 接着被复制数组中的下标后移，指向该较小元素的后继元素
  - 上述操作一直持续到两个数组中的一个被处理完为止
  - 然后在未处理完的数组中，剩下的元素被复制到新数组的尾部

Merge算法描述

```
//Merge(B[0…p-1],C[0…q-1],A[0…p+q-1])
i←0,j←0,k←0
while i<p and j<q do
   if B[i]<=C[j]
         A[k]←B[i]
         i←i+1
   else
         A[k]←C[j]
         j←j+1
   k←k+1
if i=p
   copy C[j…q-1] to A[k…p+q-1]
else
   copy B[i…p-1] to A[k…p+q-1]
```


8 3 2 9 7 1 5 4
8 3 2 9
7 1 5 4
....
8 3
2 9
7 1
5 4
8
3
2
9
7
1
5
4
3 8
2 9
1 7
4 5
2 3 8 9
1 4 5 7
1 2 3 4 5 7 8 9

例题

- 有8个关键字8, 3, 2, 9, 7, 1, 5, 4，使用归并排序方法将其排列为升序序列，给出排序过程

解：

初　始：　8，3，2，9，7，1，5，4

第一趟：　3，8，2，9，1，7，4，5

第二趟：　2，3，8，9，1，4，5，7

第三趟：　1，2，3，4，5，7，8，9

排序结束。

自然归并排序

- natural merge sort
- 进一步改进：若原始序列中存在有序子序列，则不进行分解
- [4, 8, 3, 7, 1, 5, 6, 2]
  →[4, 8], [3, 7], [1, 5, 6], [2]
  →[3, 4, 7, 8], [1, 2, 5, 6]
  →[1, 2, 3, 4, 5, 6, 7, 8]

主要内容

- 归并排序
- 快速排序
- 希尔排序
- 排序算法的分析与比较
- 关于查找的讨论

快速排序

- 思想
  - 按照元素的值进行划分
  - 对给定数组中的元素进行重新排列，以得到一个快速排序的分区
  - 在一个分区中，所有在s下标之前的元素都小于等于A[s]，所有在s下标之后的元素都大于等于A[s]
  - 建立了一个分区以后，A[s]已经位于它在有序数组中的最终位置。接下来使用同样的方法继续对A[s]前和A[s]后的子数组分别进行排序

$$\underbrace{A[0]\cdots A[s-1]}_{\text{都小于等于}A[s]}\;A[s]\;\underbrace{A[s+1]\cdots A[n-1]}_{\text{都大于等于}A[s]}$$

算法描述

```
//Quicksort[A[l…r]]
//input:数组A[0…n-1]中的子数组A[l…r]
//output:排序后的数组
if l<r
  s←Partition(A[l…r])
  Quicksort(A[l…s-1])
  Quicksort(A[s+1…r])
```

算法的前提是选择一个元素，根据该元素的值来划分子数组，这个元素就是中轴，我们暂时选择数组的第一个元素作为中轴，即p=A[l]

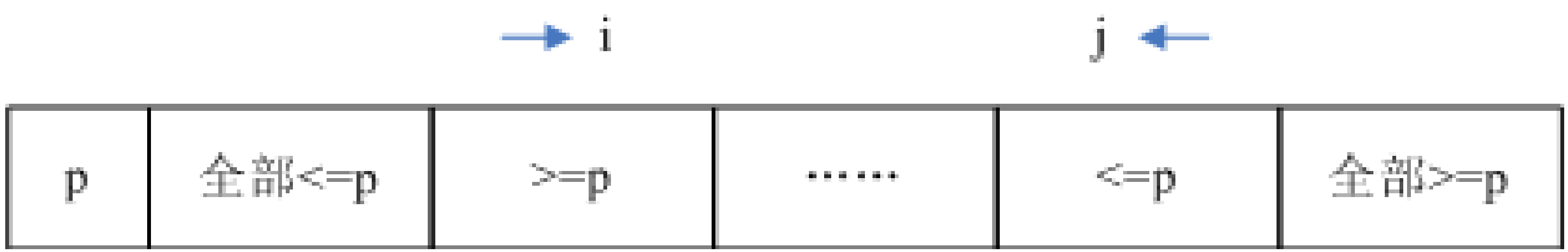
$$T(n) = O(n \log_2 n)$$

Partition算法

- 思想
  - 为了建立一个分区，有许多不同的方法对元素重新排列，其中一种是基于**两次扫描**子数组的高效算法
  - 一次是从左到右，另一次是从右到左，每次都把子数组的元素和中轴进行比较
  - 从左到右的扫描（i）从第二个元素开始，因为我们希望小于中轴的元素位于子数组的第一部分，扫描会忽略小于中轴的元素，直到遇到第一个大于等于中轴的元素才会**停止**
  - 从右到左的扫描（j）从最后一个元素开始，扫描忽略大于中轴的元素，直到遇到第一个小于等于中轴的元素才会**停止**
  - 两次扫描停止后，取决于扫描的指针是否相交，会发生3种不同的情况

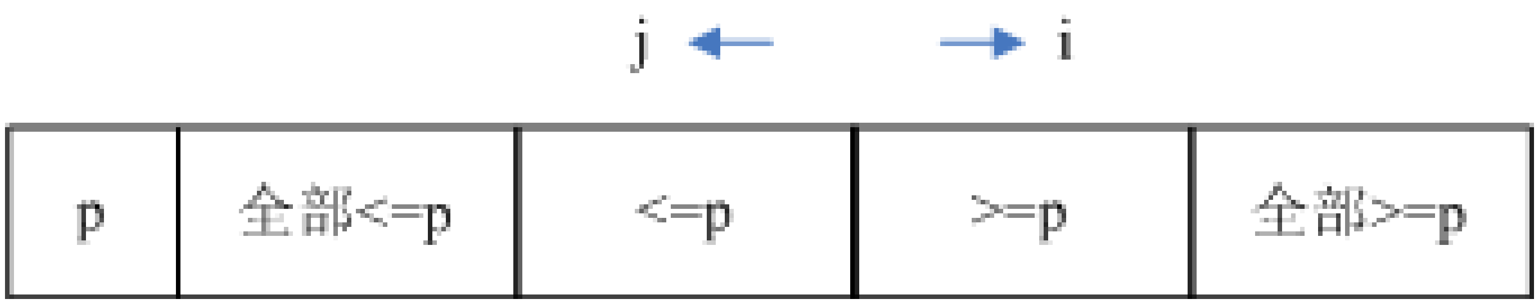
情况一

- 描述
  - 如果扫描指针i和j不相交，也就是说i<j，简单的交换A[i]和A[j]
  - 分别对i加一、j减一，然后继续开始扫描
- 示意



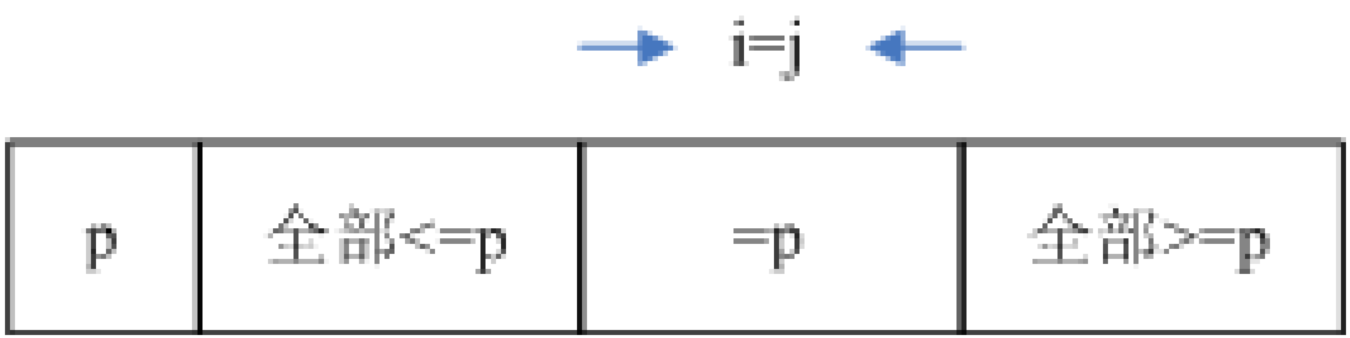
情况二

- 描述
  - 如果扫描指针相交，也就是说i>j，把中轴和A[j]交换
- 示意



情况三

- 描述
  - 如果指针停下来时指向的是同一个元素，也就是说i=j，被指向元素的值一定等于p，此时建立的分区中分裂点的位置S=i=j
- 示意



Partition算法描述

```
p←A[l]
i←l, j←r+1
repeat
  repeat i←i+1 until A[i]>=p
  repeat j←j-1 until A[j]<=p
  swap(A[i],A[j])
until i>=j
swap(A[i],A[j])
swap(A[l],A[j])
return j
```

算法合并了情况二和情况三，即当i=j时也做了一次无谓的交换

i可能越界而j不可能越界，因此实现时需要对i进行特殊处理

0	1	2	3	4	5	6	7
	i						*j*
5	3	1	9	8	2	4	7
			i			*j*	
5	3	1	9	8	2	4	7
			i			*j*	
5	3	1	4	8	2	9	7
				i	*j*		
5	3	1	4	8	2	9	7
				i	*j*		
5	3	1	4	2	8	9	7
				j	*i*		
5	3	1	4	2	8	9	7
2	3	1	4	$\boxed{5}$	8	9	7
	i		*j*				
2	3	1	4				
	i	*j*					
2	3	1	4				
	i	*j*					
2	1	3	4				
	j	*i*					
2	1	3	4				
1	$\boxed{2}$	3	4				
1							
			i j				
		3	4				
		j	*i*				
		$\boxed{3}$	4				
			4				
						i	*j*
					8	9	7
						i	*j*
					8	7	9
						j	*i*
					8	7	9
					7	$\boxed{8}$	9
					7		
							9

快速排序的改进

- 改进
  - 随机数、两平均、三平均中轴选择算法
  - 当子数组足够小时改用最简单的排序算法
  - 综合运用这些措施，可缩减20%时间

提示

- 快速排序算法要求熟练掌握源代码

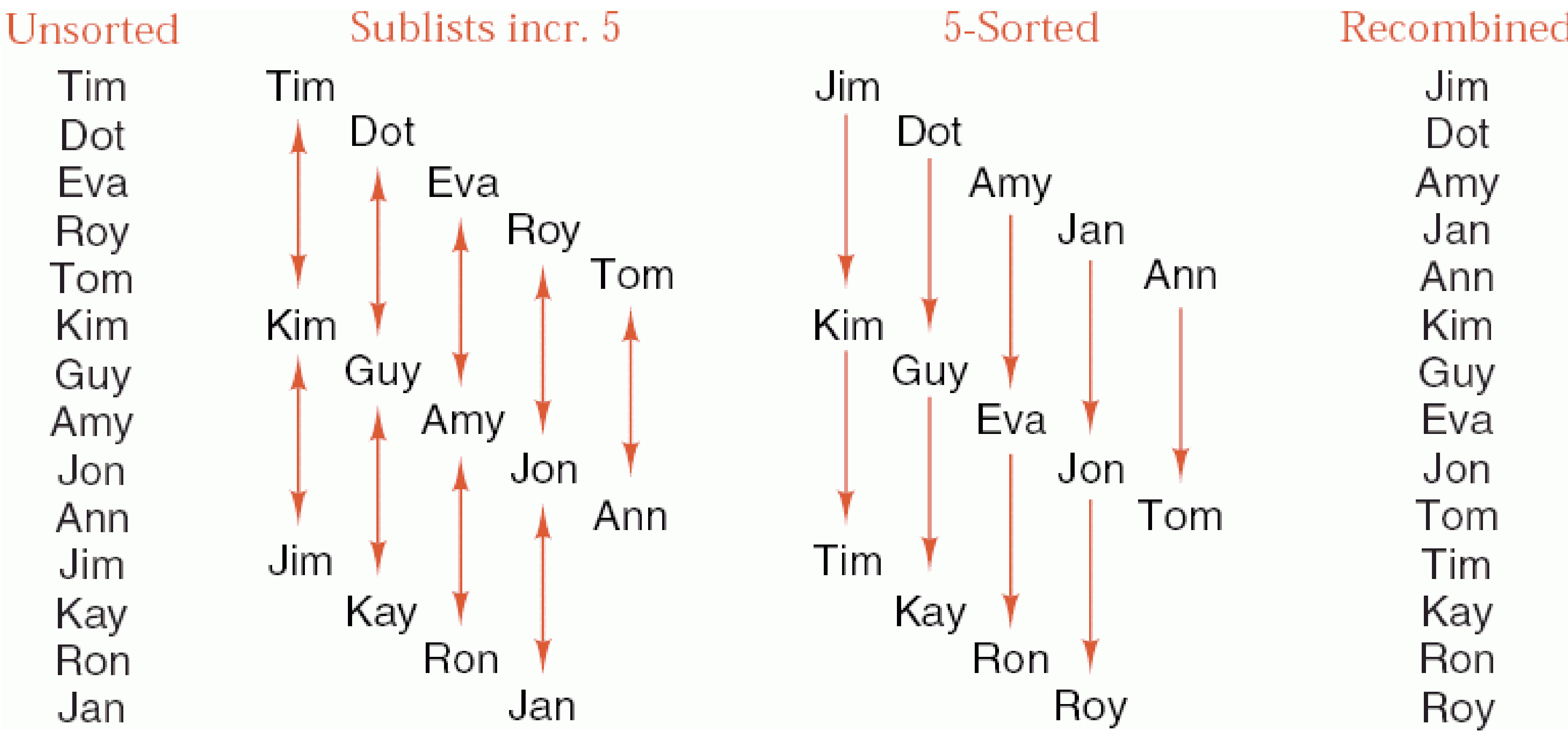
主要内容

- 归并排序
- 快速排序
- 希尔排序
- 排序算法的分析与比较
- 关于查找的讨论

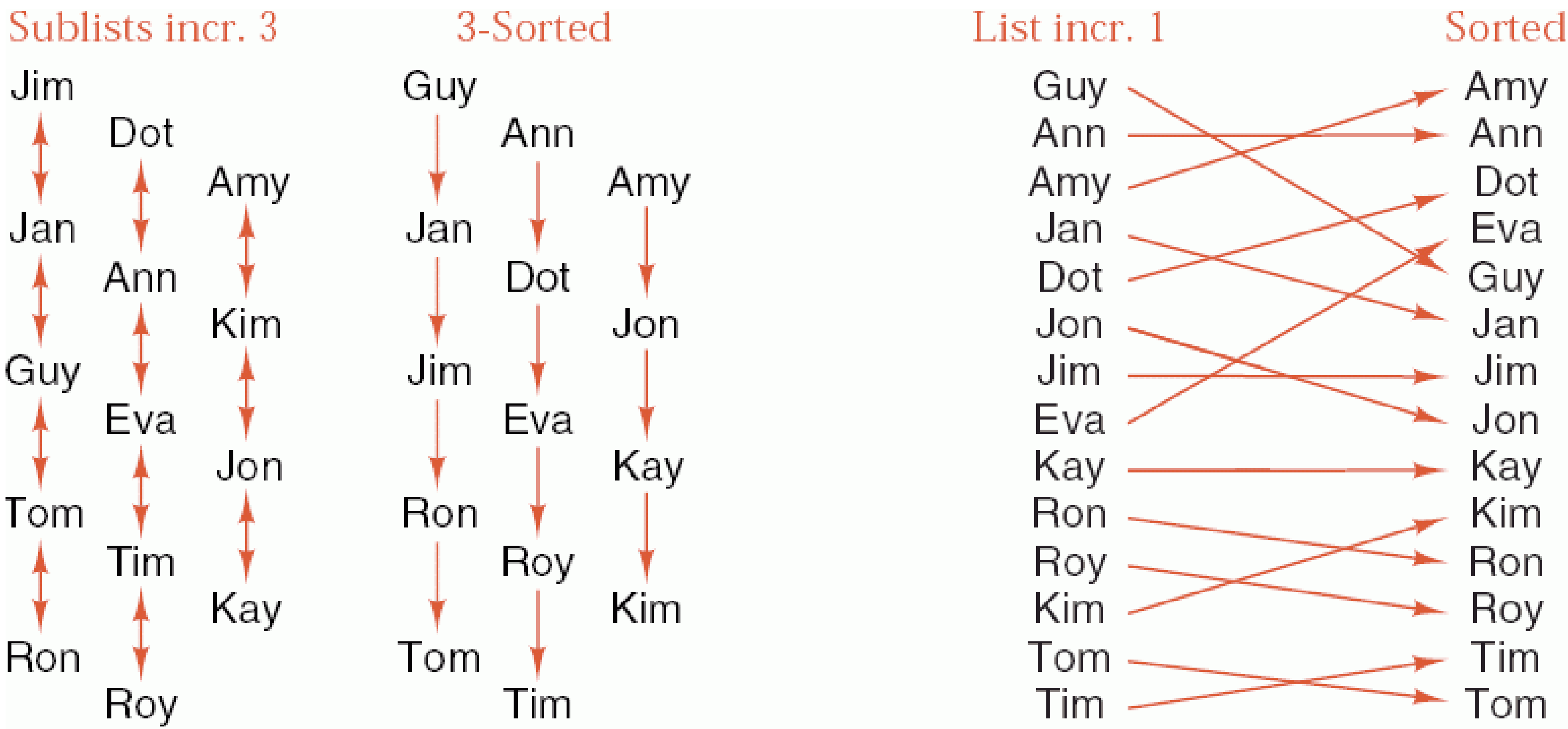
希尔排序

- 又叫缩小增量排序
- 算法思想：设待排序列含n个元素
  - 取整数gap=floor(n/3)+1，将每隔gap的元素放在一个子序列中，对子序列插入排序
  - 然后缩小间隔，令gap=floor(gap/3)+1，对新的子序列插入排序
  - 重复上述过程，直至gap=1时最后执行一次

Shell排序例


Unsorted
Tim
Dot
Eva
Roy
Tom
Kim
Guy
Amy
Jon
Ann
Jim
Kay
Ron
Jan
Sublists incr. 5
Tim
Dot
Eva
Roy
Tom
Kim
Guy
Amy
Jon
Ann
Jim
Kay
Ron
Jan
5-Sorted
Jim
Dot
Amy
Jan
Ann
Kim
Guy
Eva
Jon
Tom
Tim
Kay
Ron
Roy
Recombined
Jim
Dot
Amy
Jan
Ann
Kim
Guy
Eva
Jon
Tom
Tim
Kay
Ron
Roy

Shell排序例（续）



Sublists incr. 3
3-Sorted
List incr. 1
Sorted
Jim
Dot
Amy
Jan
Ann
Kim
Guy
Eva
Jon
Tom
Tim
Kay
Ron
Roy
Guy
Ann
Amy
Jan
Dot
Jon
Jim
Eva
Kay
Ron
Roy
Kim
Tom
Tim
Guy
Ann
Amy
Jan
Dot
Jon
Jim
Eva
Kay
Ron
Roy
Kim
Tom
Tim
Amy
Ann
Dot
Eva
Guy
Jan
Jim
Jon
Kay
Kim
Ron
Roy
Tim
Tom

算法分析

- 开始时
  - 序列数较多，每个序列元素数较少，排序快
- 排序后期
  - 子序列元素数增多，但大多有序，再执行插入排序效率较高

- 希尔排序性能与gap选取有关，一般认为其优于O(n^2)，但很难达到O(nlogn)

实现

```
template<class T>
void ShellSort(T a[], int n)
{
  int increment, start;

  for (increment = n / 3 + 1; increment >= 1;
                                increment = increment / 3 +1)
        for (start = 0; start < increment; start++)
                insertsort_interval(a, n, start,
  increment);
}
```

实现（续）

```
template<class T>
void insertsort_interval(T a[], int n, int start,
                                        int increment)
{
  for (int i = start + increment; i < n; i += increment)
  {
        T t = a[i];
        for (int j = i – increment; j >= start && t < a[j];
                                   j -= increment)
              a[j + increment] = a[j];
        a[j + increment] = t;
  }
}
```

主要内容

- 归并排序
- 快速排序
- 希尔排序
- **排序算法的分析与比较**
- 关于查找的讨论

时间复杂度比较

方　法	最坏复杂性	平均复杂性
冒泡排序	n^2	n^2
计数排序	n^2	n^2
插入排序	n^2	n^2
选择排序	n^2	n^2
堆排序	$n\log n$	$n\log n$
归并排序	$n\log n$	$n\log n$
快速排序	$\boxed{n^2}$	$n\log n$

初始序列有序时

部分结论

（1）平均来看，快排最优，但若初始序列有序，则快排性能降至n^2级。使用三平均选中轴法可避免最差情况，结合插入排序效果更好。

（2）插入、选择、冒泡都属“简单排序”，复杂度是n^2，它们中间当待排序列基本有序或n较小时，插入排序更好。

部分结论

（3）稳定排序算法有：插入、冒泡、归并、基数

（4）不稳定排序算法有：简单选择、希尔、快速、堆排序

主要内容

- 归并排序
- 快速排序
- 希尔排序
- 排序算法的分析与比较
- 关于查找的讨论

顺序查找

- 又叫线性查找
- 从表头开始，依次比较关键值，直到找到为止，记为成功
- 如果整个表都查完仍未找到，则记为失败
- 注意：一般通过设置“监视哨”来提高顺序查找效率

折半查找

- 又叫二分查找
- 要求待查序列有序
- 先比较最中间的那个元素，如果不匹配，选择在左侧区间或右侧区间继续
- 优势是每比较一次，查找区间缩小一半
- 复杂度为0(logn)

并查集算法

- 一种基于集合的特殊查找
  - 处理不相交集合的合并及查询问题；
  - 常常以森林来表示。
- 举例
  - 11个元素（1~11）
  - 10种相关关系（1&2，3&4，2&5，4&6，2&6，7&11，7&8，7&9，9&11，1&6）
  - 假设相关关系有传递性，则该例中不相交集合有几个？

并查集算法

（1&2，3&4，2&5，4&6，2&6，7&11，7&8，7&9，9&11，1&6）

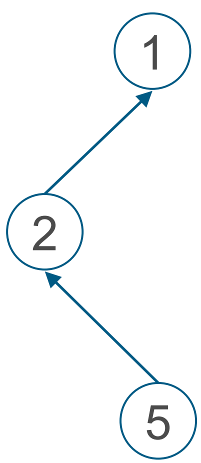
1	2	3	4	5	6	7	8	9	10	11
1	2	3	4	5	6	7	8	9	10	11

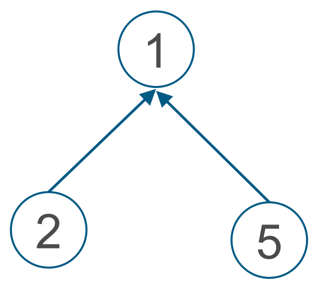
1	1	3	3	5	6	7	8	9	10	11
1	2	3	4	5	6	7	8	9	10	11

并查集算法

（1&2，3&4，2&5，4&6，2&6，7&11，7&8，7&9，9&11，1&6）

1	1	3	3	5	6	7	8	9	10	11
1	2	3	4	5	6	7	8	9	10	11



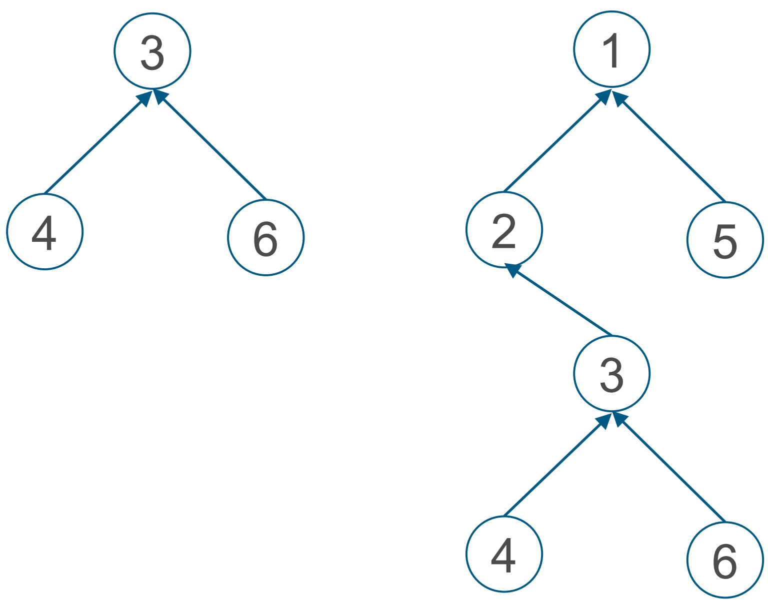


1	1	3	3	1	6	7	8	9	10	11
1	2	3	4	5	6	7	8	9	10	11

并查集算法

（1&2，3&4，2&5，4&6，2&6，7&11，7&8，7&9，9&11，1&6）

1	1	3	3	1	6	7	8	9	10	11
1	2	3	4	5	6	7	8	9	10	11

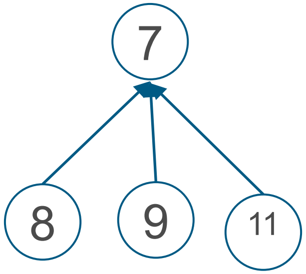


1	1	1	1	1	1	7	8	9	10	11
1	2	3	4	5	6	7	8	9	10	11

并查集算法

（1&2，3&4，2&5，4&6，2&6，7&11，7&8，7&9，9&11，1&6）

1	1	1	1	1	1	7	8	9	10	11
1	2	3	4	5	6	7	8	9	10	11

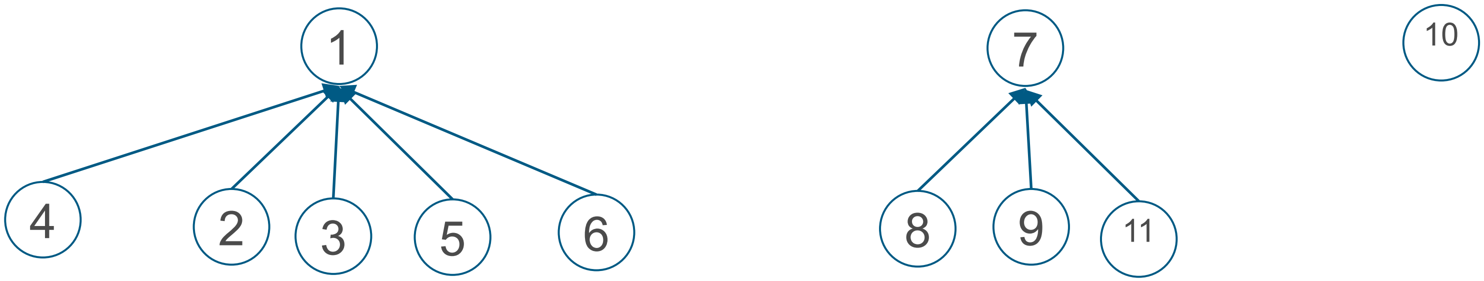


1	1	1	1	1	1	7	7	7	10	7
1	2	3	4	5	6	7	8	9	10	11

并查集算法

（1&2，3&4，2&5，4&6，2&6，7&11，7&8，7&9，9&11，1&6）

1	1	1	1	1	1	7	7	7	10	7
1	2	3	4	5	6	7	8	9	10	11



集合查找结果：
{1,2,3,4,5,6}
{7,8,9,11}
{10}

本章结束