华中科技大学

数据结构

第10章　内部排序

主讲教师：周时阳

内容摘要

《数据结构》是计算机科学与技术类各专业的一门基础课。

本章主要介绍数据结构课程研究的问题背景、研究内容和范围。讨论了数据结构和算法的基本概念以及算法的评价。

关于线性结构、树型结构和图型结构等3类基本结构，将在后续各章陆续展开讨论它们的逻辑结构、逻辑结构上定义的运算、物理结构、逻辑结构与物理结构对应关系、运算的实现算法与效率分析。

重点讲解

小结

10.1 概论

“排序”是基于数据逻辑结构(D,R)定义的一种十分常见的运算。数学上，“排序” 是指依据D中的每个数据元素之关键字，按照递增(或递减)顺序将数据元素排列的过程。即：

假设 $D=\{a_1, a_2, \ldots, a_i, \ldots, a_n\}$初始序列为

$(a_1, a_2, \ldots, a_i, \ldots, a_n)$

其对应的关键字序列为

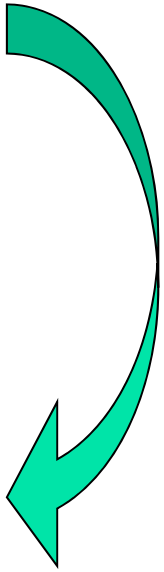
$(k_1, k_2, \ldots, k_i, \ldots, k_n)$

关键字的递增序列为

$(k_{p1}, k_{p2}, \ldots, k_{pi}, \ldots, k_{pn})$

则 (a1, a2, …, ai, …, an)排序结果为

$(a_{p1}, a_{p2}, \ldots, a_{pi}, \ldots, a_{pn})$



目录

如果排序算法，对于次关键字排序后，确保相同的次关键字与排序前的相对前后次序不变，则称该算法是稳定的，否则称该算法是不稳定的。即：

假设D的初始序列为

$$(\ldots, a_i, \ldots, a_j, \ldots)$$

其对应的关键字序列为

$$(\ldots, k_i, \ldots, k_j, \ldots)$$

且$k_i=k_j$，关键字的递增序列为

$$(\ldots, k_i, k_j, \ldots)$$

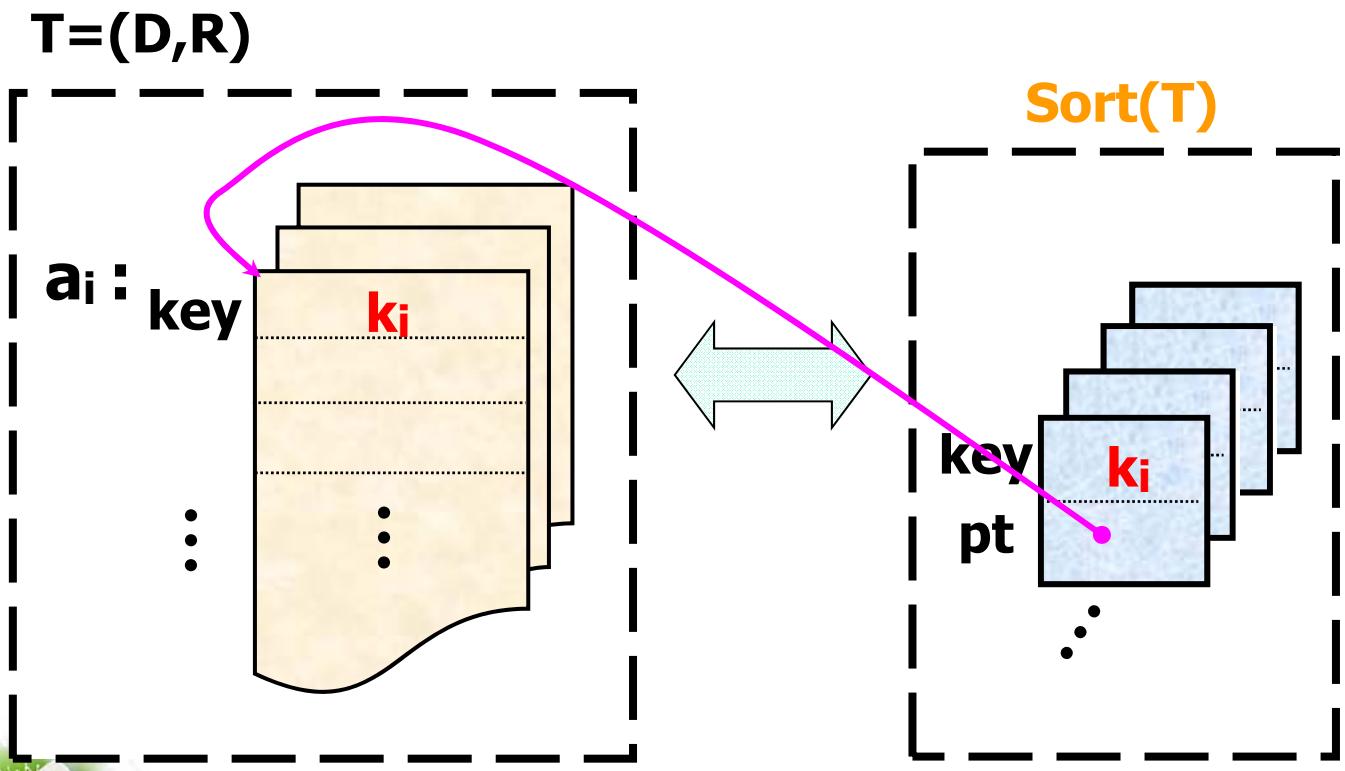
相对次序不变！

则 D的排序结果为

$$(\ldots, a_i, a_j, \ldots)$$

标识多个(>1)数据元素的分量，称为次关键字。

“排序”运算的一般描述形式为：sort(T,f_{key},up_dn)，即根据给定关键字分量，按照up_dn指定的不减序或不增序，对T进行排序。



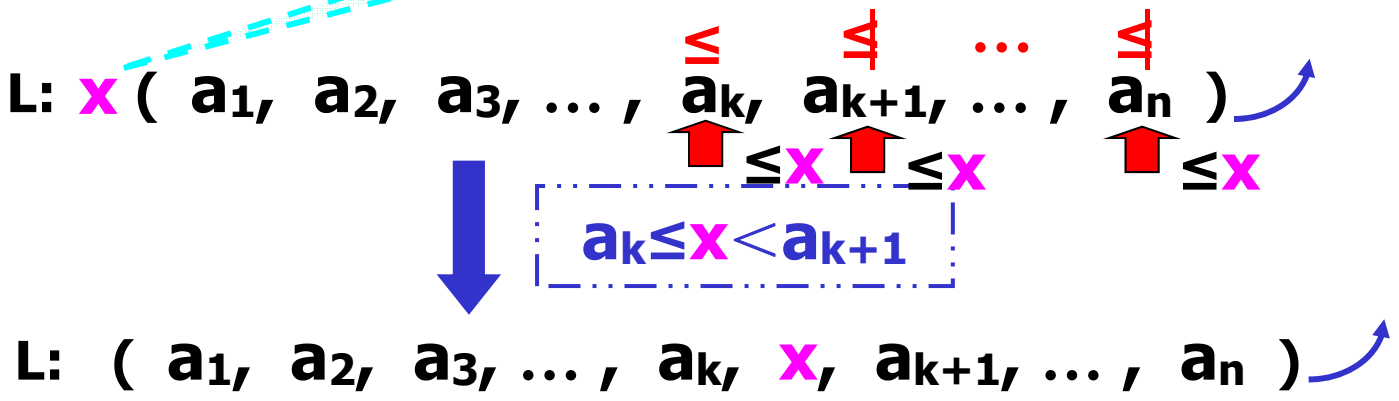
10.2 插入排序

10.2.1 直接插入排序

“哨兵技术”

基本原理：

在递增(递减)有序表L上，插入一个元素x，使其仍然保持有序。



基本步骤：

(1) 确定插入位置；

(2) 移动元素；

(3) 填入新元素。

基本步骤：

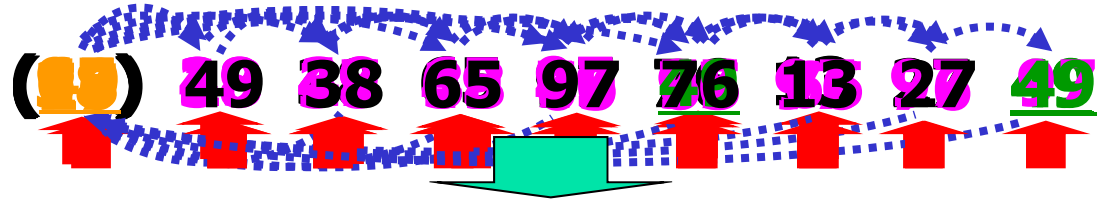
(1) 边确定插入位置，边移动元素；

(2) 填入新元素。

目录

算法思想：

从表$L=(a_1, a_2, a_3, \ldots, a_i, \ldots, a_n)$的第2个元素$a_2$开始，直到最后一个元素$a_n$为止，逐个插入本元素之左边的有序子表，使其仍然保持有序。



算法分析：(问题规模为参加排序的元素个数n)

最好情况：正序

比较次数：$T_b(n)= n-1$　　移动次数：$T_b()=0$

$T_w(n)=\sum_{i=2}^{n} i$　　$T_w()=\sum_{i=2}^{n}(i+1)$

$T_a(n) = (T_b+T_w)/2 \approx n^2/4$

最差情况：反序

算法时间复杂度$T(n)=O(n^2)$

10.2.2 其它插入排序

1. 折半插入排序

算法思想：

“定位”除了上述顺序定位法外，还可以利用有序子表特点，采用折半定位法。

$L: (a_1, a_2, a_3, \ldots, a_{k-1}, a_k, a_{k+1}, \ldots, a_{i-1})$　　a_i

$x < a_k$　　$a_k \le x$

H　L　H　L

当定位结束(即L>H)时，$a_L \sim a_{i-1}$向后移动。

算法分析：

比较次数：$T_b(n) = n-1$　　移动次数：不变

$$T_w(n) = \sum_{i=1}^{n-1} \log i \approx n \log n$$

算法时间复杂度$T(n)=O(n^2)$

2. 2路插入排序 (将关键字排序在辅助空间中)

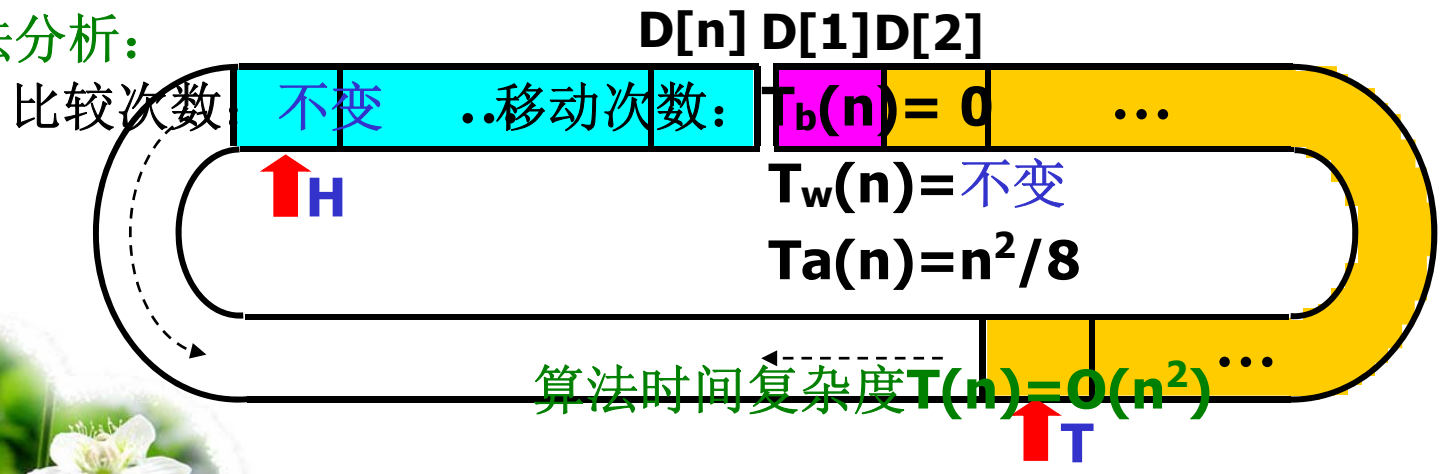
算法思想：

⑴ 取出表 $L=(a_1, a_2, a_3, \ldots, a_i, \ldots, a_n)$ 的第1个元素 a_1 存入辅助空间D[1]；

⑵ 取出表L的下一个元素 a_i，如果a_i＜D[1]，则插入到D[1]之左边的有序子表，否则插入到D[1]之右边的有序子表；

⑶ 重复⑵，直到表L的最后一个元素a_n插入后为止。

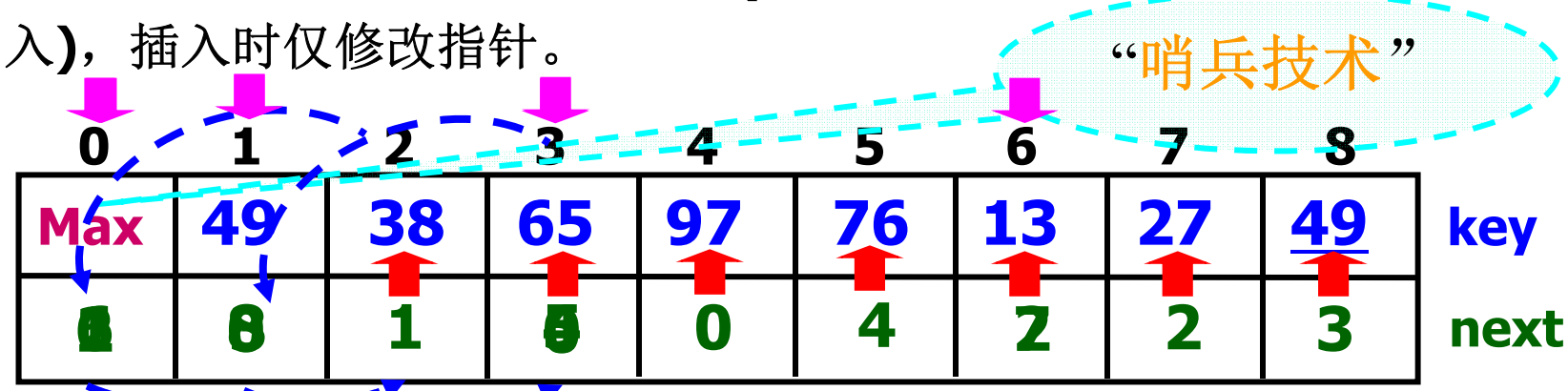
算法分析：



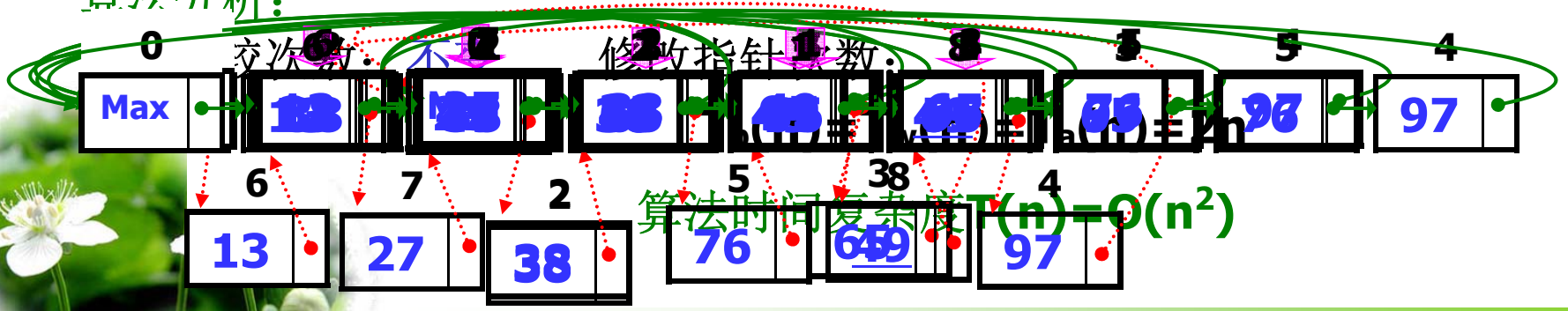
3. 表插入排序 (静态链表，a_i内部结构增加一个静态指针)

算法思想：

从表$L=(a_1, a_2, a_3, \ldots, a_i, \ldots, a_n)$的第2个元素 a_2 开始，直到最后一个元素 a_n 为止，逐个插入本元素之左边的有序子表，使其仍然保持有序。从小到大方向定位 (即第一次遇到大于a_i的元素之前插入)，插入时仅修改指针。



算法分析：



交换次数：不定

修改指针次数：$n-1$

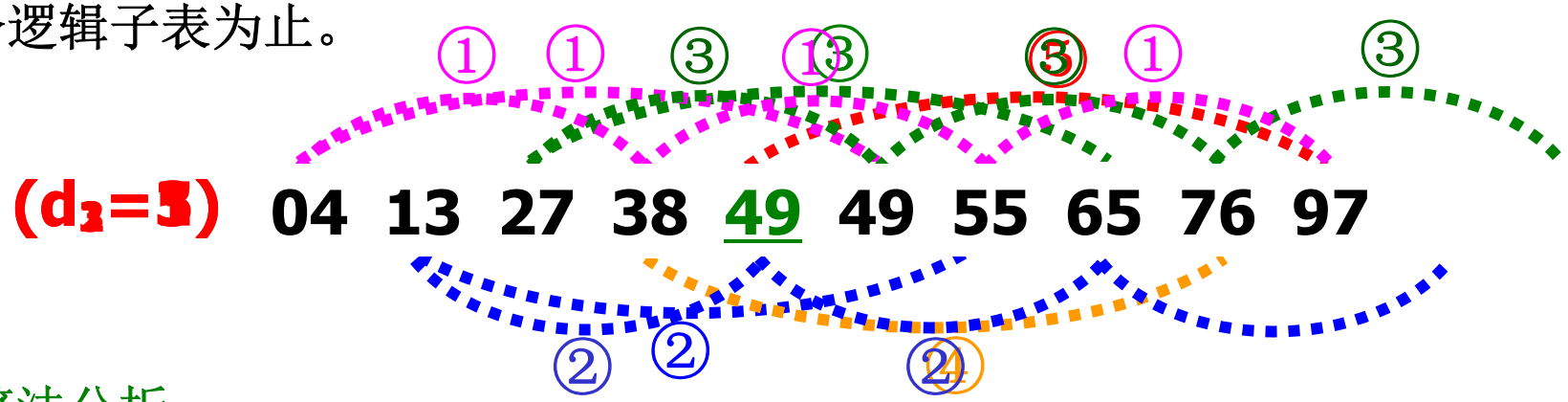
算法时间复杂度$T(n)=O(n^2)$

10.2.3 希尔排序

算法思想：

⑴ 将表$L=(a_1, a_2, a_3, \ldots, a_i, \ldots, a_n)$划分成若干逻辑子表，分别对其进行直接插入排序；

⑵ 依据逻辑子表个数逐步递减原则，重复⑴，直到表L构成一个逻辑子表为止。



算法分析：

希尔排序的时间复杂性问题，与增量序列密切相关，至今仍然没有定论。目前已经公布的结果有：$O(n^{1.5})$和$O(n^{1.3})$。

10.3 快速排序

基本原理：

通过两个元素之间交换，逐步使得元素移动到表L的正确位置。

1. 冒泡排序

算法思想：

⑴ 对于表$L=(a_1, a_2, a_3, \ldots, a_i, \ldots, a_n)$，从左至右的顺序，将序号相邻的、反序的两个元素交换之；

⑵ 对于除最后一个之外的剩余部分构成的子表重复⑴，直到剩余部分构成的子表表长等于1为止。

算法分析：

比较次数：

$$T_b(n)=(n-1)+(n-2)+\cdots+1=O(n^2)$$

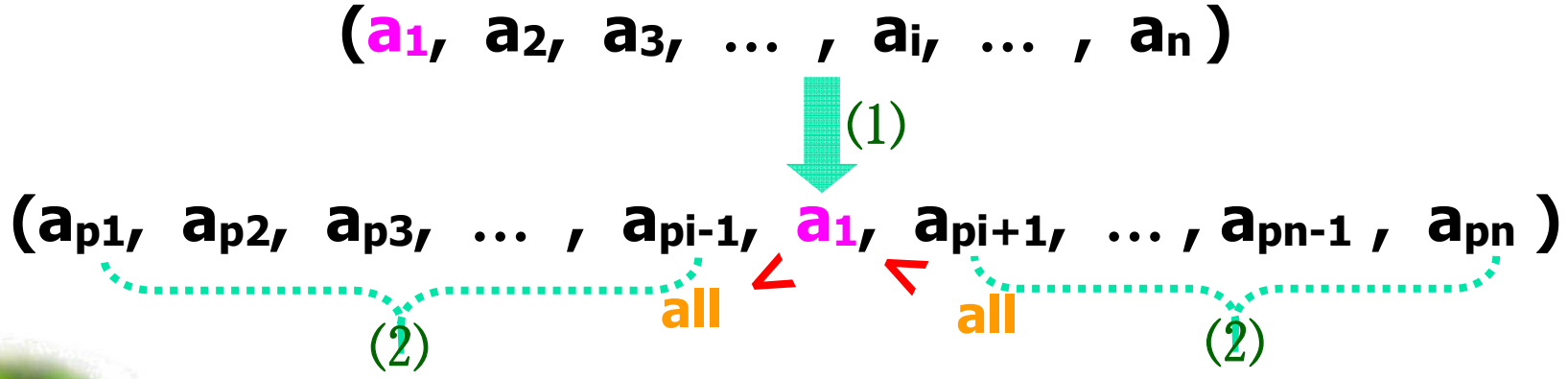
$$T_w(n)=T_b(n)=T_a(n)=O(n^2)$$

2. 快速排序

算法思想：

⑴ 依据表L=(a_1, a_2, a_3, … , a_i, … , a_n)的第一个元素a_1，将取L“划分”成左右2个逻辑子表，使得a_1小于左子表的所有元素，且大于右子表的所有元素；

⑵ 左右子表分别递归处理。



显然，算法的核心是“划分”处理！

划分方法：(low和high初始值分别指向L之最左和最右的单元)

(1) 当low<high时，重复做如下处理：

(1.1) 向左移动high，将首次遇到的小于L[low]之L[high]（<）与L[low]交换；

(1.2) 向右移动low，将首次遇到的大于L[high]之L[low]（≥）与L[high]交换；

算法分析：

设排序元素个数为n，统计比较次数，则

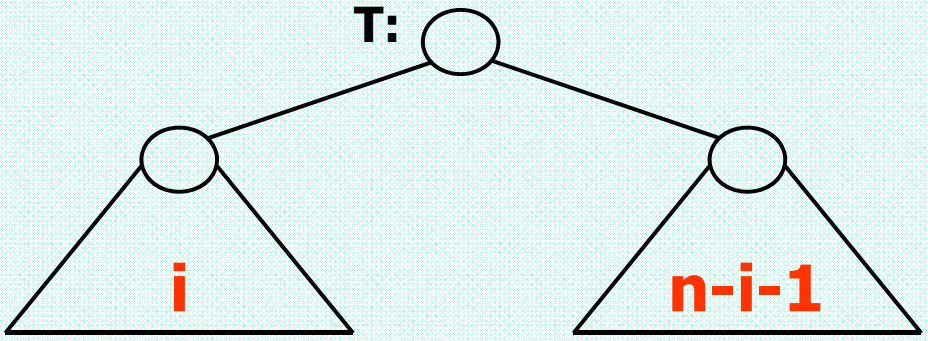
$T_w(n) \le n^2$ {在有序情况下}

$T_b(n) \le n \log n$ {在每次划分均为左、右子表相等情况下}

$T_a(n) \le cn \log n$

假设n个元素的平均比较次数为$T_a(n)$，划分后左子表元素个数为i，并且i取(0~n-1)是等概率的，则

$$T_a(n) = dn + \frac{1}{n}\sum_{i=0}^{n-1}(T_a(i)+T_a(n-i-1)) \le cn \log n$$



10.4 选择排序

基本原理：

选择表L的最大(小)元素，与最后位置上元素交换。

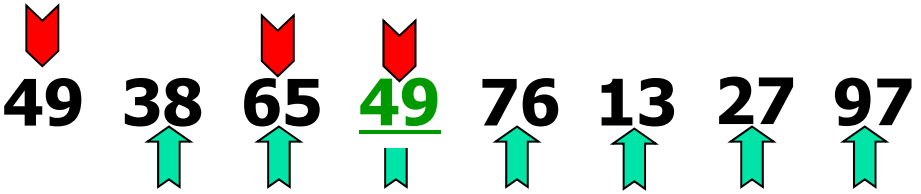
10.4.1 简单选择排序

算法思想：

⑴ 对于表 $L=(a_1, a_2, a_3, \ldots, a_i, \ldots, a_n)$，顺序遍历，选择出最大元素所在位置，与最后位置元素交换；

⑵ 对于除最后一个元素之外的剩余部分构成的子表重复⑴，直到剩余部分构成的子表表长等于1为止。

算法分析：



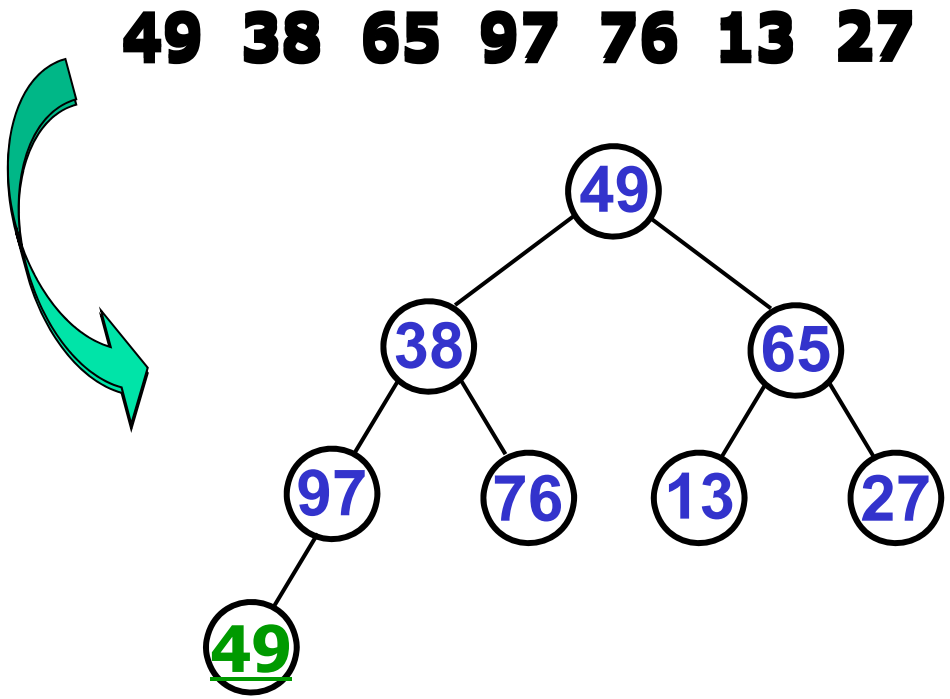
比较次数：

$$T_w(n)=T_b(n)=(n-1)+(n-2)+\ldots+(1)=O(n^2)=T_a(n)$$

目录

10.4.3 堆排序

表$L=(a_1, a_2, a_3, \ldots, a_i, \ldots, a_n)$，可以视同是完全二叉树的顺序存储结构，其一一对应关系有二叉树的性质5决定。

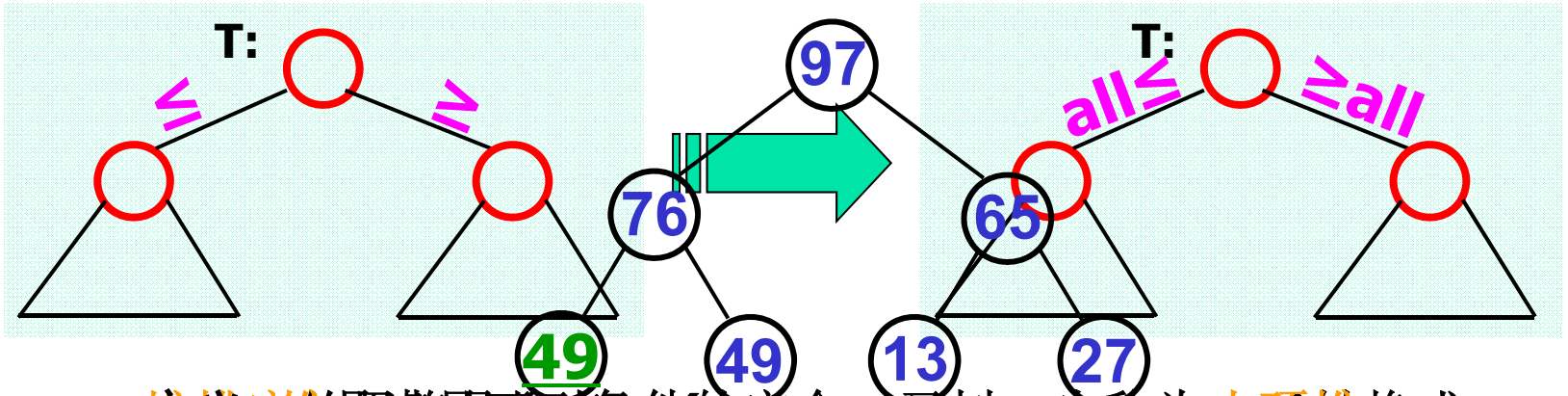


定义 堆是满足下列条件的完全二叉树，也称为大顶堆。

(1) T的根值不小于其左子树和右子树的根值；

(2) 左子树和右子树均满足(1)。

特别约定，空完全二叉树为大顶堆，且根值为 $-\infty$ 。



定义 堆是满足下列条件的完全二叉树，也称为小顶堆。

(1) T的根值不大于其左子树和右子树的根值；

(2) 左子树和右子树均满足(1)。

特别约定，空完全二叉树为小顶堆，且根值为 $+\infty$ 。

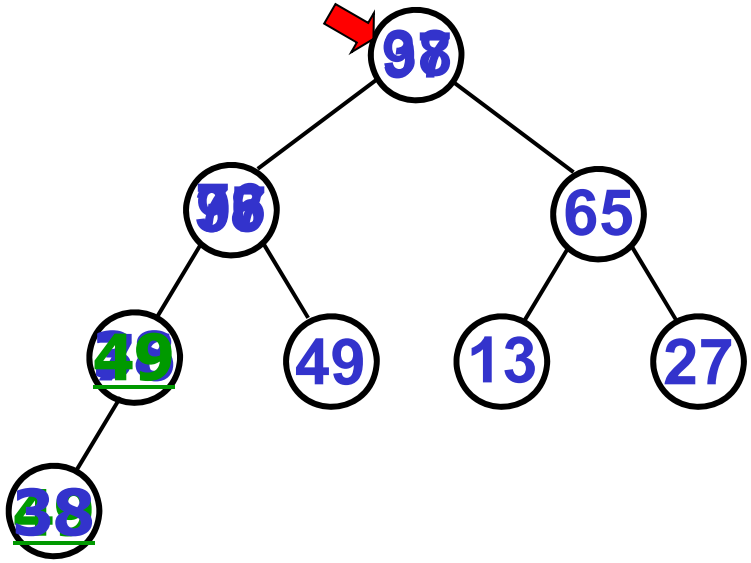
堆排序的思路是将序列 $K=(a_1, a_2, a_3, \ldots, a_n)$ 转换成大顶堆，将堆的根（即最后一个位置的元素）与最后一个叶子外剩余部分再重复上述处理。

假设表$L=(a_1, a_2, a_3, \ldots, a_i, \ldots, a_n)$对应的完全二叉树T是左右子树为堆、仅仅是T的根不满足堆的条件之情况，则将T转换成堆的过程称为调整堆。

调整堆算法思想：

⑴ 将树根与其左右子树根值最大者交换；

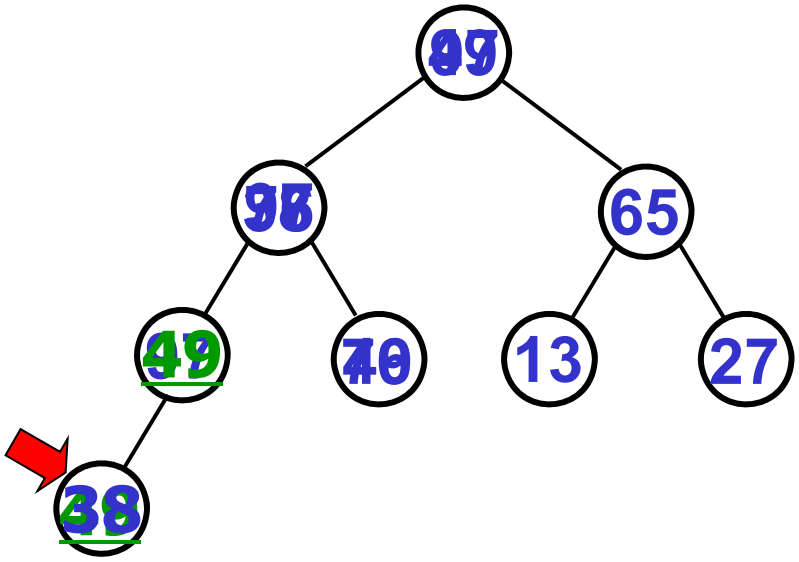
⑵ 对交换后的左(或右)子树重复⑴，直到左(或右)子树为堆。



创建堆算法思想：

从最大序号开始逐步到根，对于每个子树，采用调整堆算法使其成堆。

97 38 65 49 49 13 27 38



注：从最后一个非叶子结点开始即可！

堆排序算法思想：(n — 表长)

⑴ 创建堆：从最后一个非叶子结点开始逐步到树根，对于每个子树进行调整堆；

⑵ 重复**n-1** 次如下处理：将堆的根与最后一个叶子交换；除最后一个叶子之外剩余部分再调整堆。

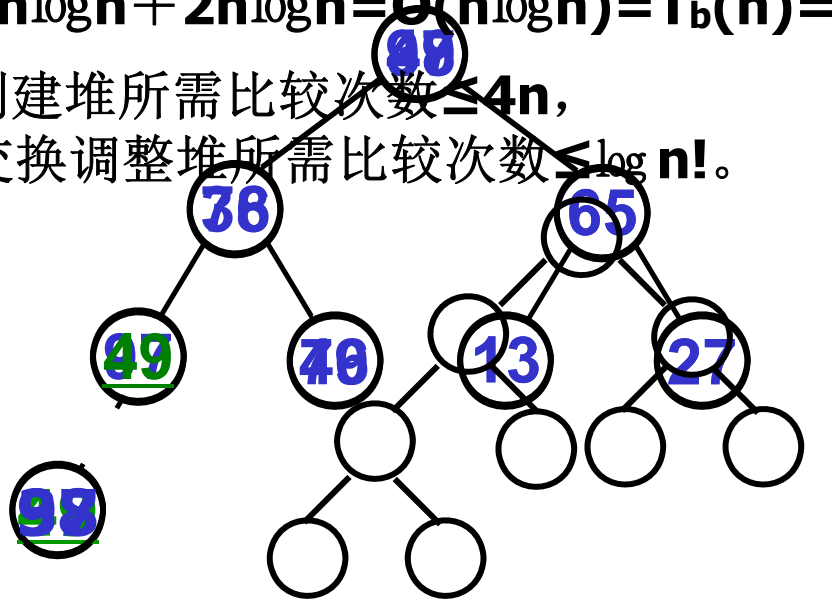
算法分析：(比较次数)

25 36 65 49 40 13 27 97

$$T_w(n) \le 2n\log n + 2n\log n = O(n\log n) = T_b(n) = T_a(n)$$

实际上，⑴ 创建堆所需比较次数≤4n，

⑵ 交换调整堆所需比较次数≤$\log n!$。



10.5 归并排序

基本原理：

两个有序表和合并(merge)成为一个有序表。(算法见例2-2)

算法思想：

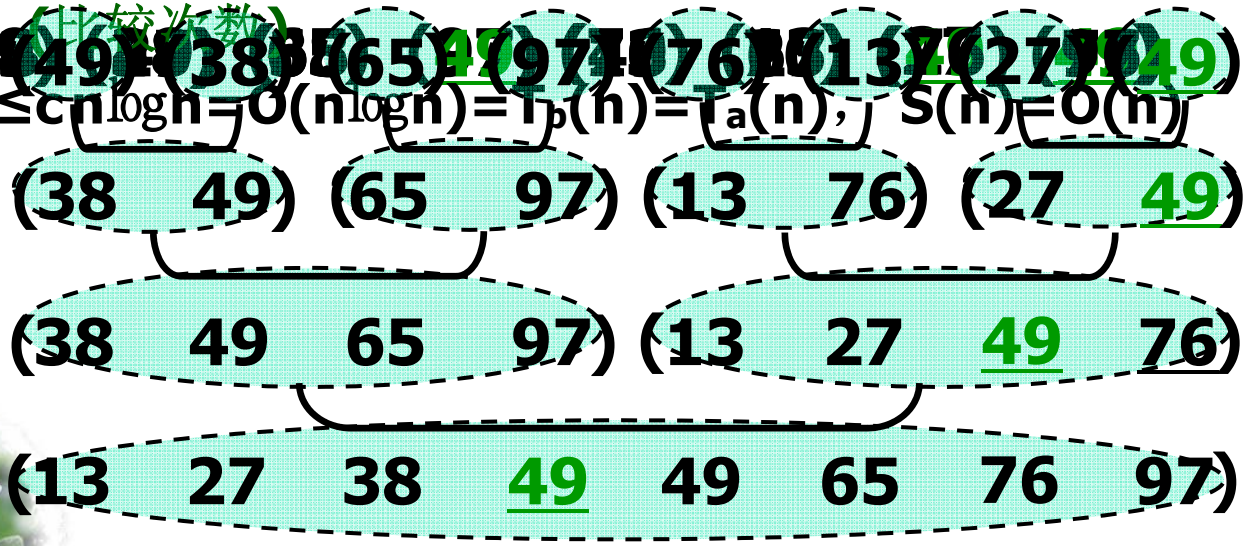
(L=(a1, a2, a3, … , ai, … , an)的每个元素，看成一个有序子表)

⑴ 从左至右，将相邻的两个有序子表合并之；

⑵ 重复⑴，直到所有子表合并成一个有序子表为止。

算法分析：(比较次数)

$T_w(n) \le cn\log n = O(n\log n) = T_b(n) = T_a(n)$，$S(n) = O(n)$



目录

10.6 基数排序

10.6.1 多关键字排序

最高位优先排序 (MSD--Most Significant Digit first)

依据第1关键字K^0排序；之后，对于具有相同K^0值的子序列，再依据第2关键字K^1分别排序；一次类推直到对相应子序列分别依据第d关键字K^d排序为止，将所有子序列链接在一起构成整个有序序列。

最低位优先排序 (LSD--Least Significant Digit first)

依据第d关键字K^d排序；之后，对于整个序列，再依据第d-1关键字K^{d-1}排序；一次类推直到对相应子序列分别依据第1关键字K^1排序为止，整个有序序列。

(3,D) (2,B) (3,B) (1,C) (3,C) (2,C) (3,A) (1,B) (3,E) (2,A) (1,A) (2,B)

(1,C) (1,B) (1,A) (2,D) (2,C) (2,A) (2,B) (3,D) (3,B) (3,C) (3,A) (3,E)

(3,A) (2,A) (1,A) (3,B) (1,B) (2,B) (1,C) (2,C) (3,D) (2,D) (3,C) (3,E)

(1,A) (1,B) (1,C) (2,A) (2,B) (2,C) (2,D) (3,A) (3,B) (3,C) (3,D) (3,E)

此方法要求采用的排序算法必须具有稳定性。

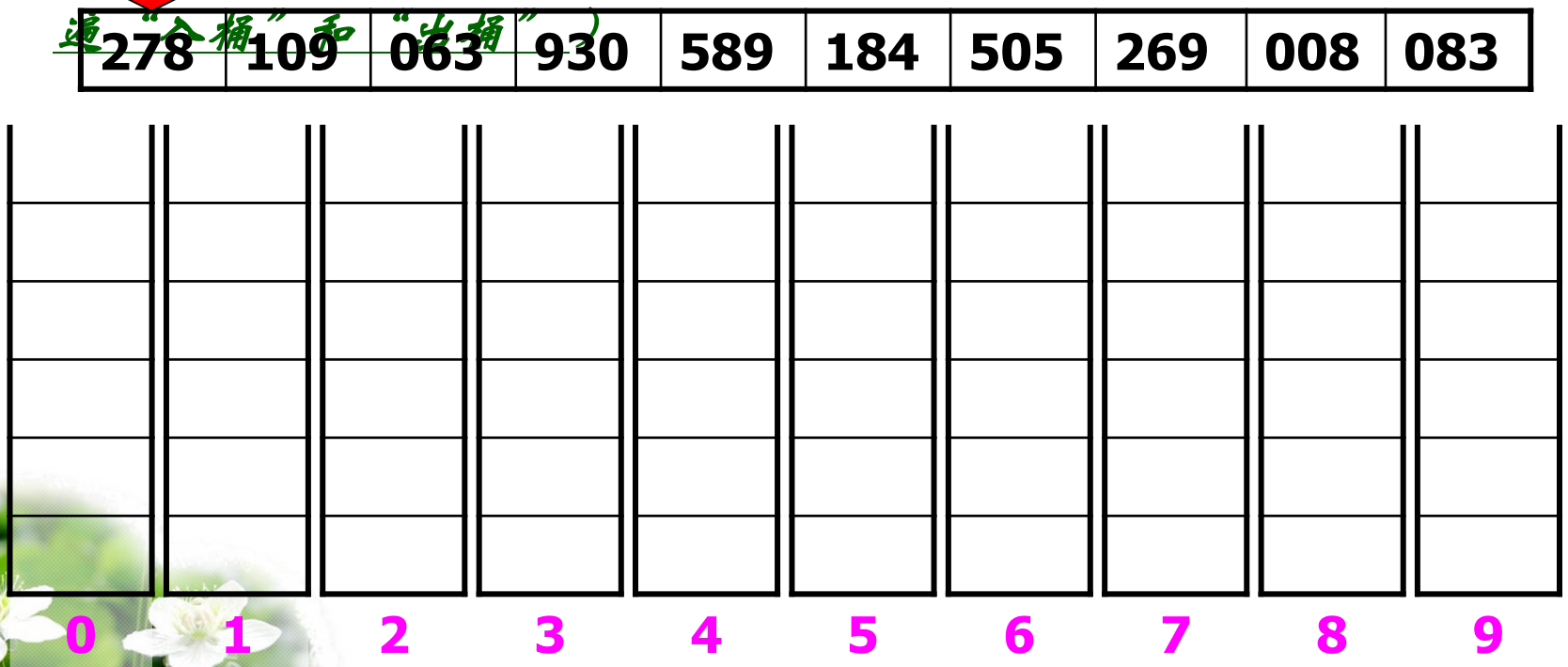
提示：在特殊情况下，可以将多关键排序问题转换成单关键字排序问题，即一次排序完成。

目录

10.6.2 基数排序

基数排序的原理是将关键字每一位K^m视同单关键字，这样就构成多关键字$(K^0\ K^1\ K^2\ \ldots\ K^{n-1})$排序，并采用LSD方法进行排序。

基数排序的算法思想是按照每位K^m $(m=n-1\sim0)$值，将关键字依次进行“入桶”操作，以后依次进行“出桶”操作。（即重复n遍“入桶”和“出桶”）



10.7 内部排序方法的比较

方法排序		Ta()	Tw()	S()
插入	直接插入排序			
	折半插入排序			
	2-路插入排序			
	表插入排序			
	希尔排序			
交换	冒泡排序			
	快速排序			
选择	简单选择排序			
	树形选择排序			
	堆排序			
	归并排序			
	基数排序			

目录

小结

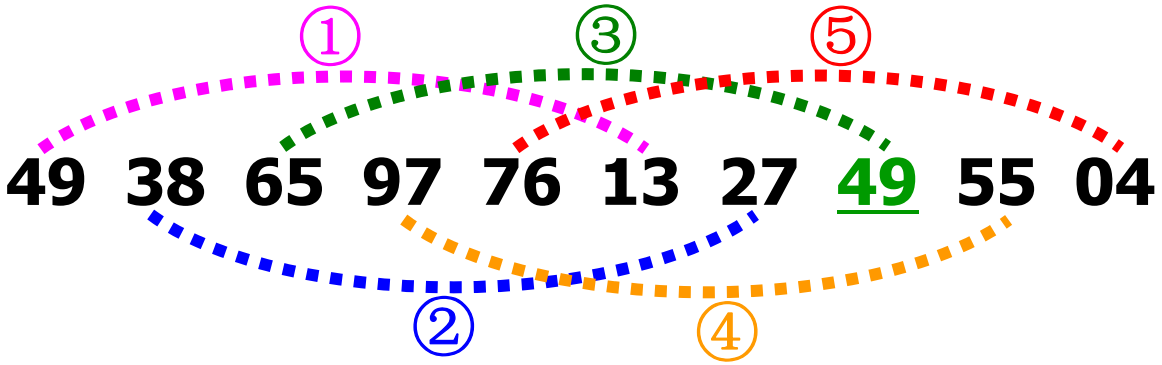
本章重点介绍了数据结构研究对象、内容和方法，并重点讨论了数据元素存储结构和算法效率估算方法。

数据逻辑结构基本类型划分为集合结构、线性结构、树形结构和图状结构。

研究的主要内容是（**1**）数据的逻辑结构、（**2**）逻辑结构上定义的运算、（**3**）数据的物理结构、（**4**）逻辑结构与物理结构的对应关系和（**5**）运算基于物理结构的实现算法及效率分析。

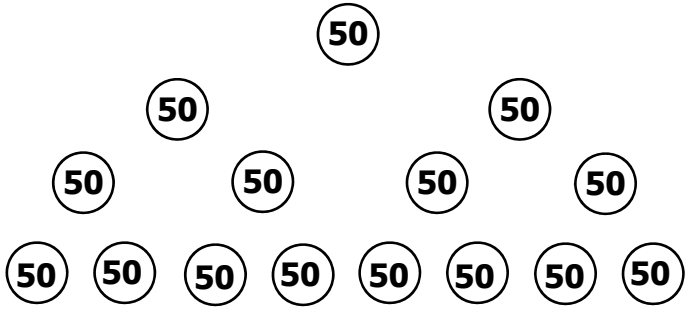
提出的基本概念是数据、数据元素、数据对象、逻辑结构、关系、物理结构、数据类型、算法和复杂度等。

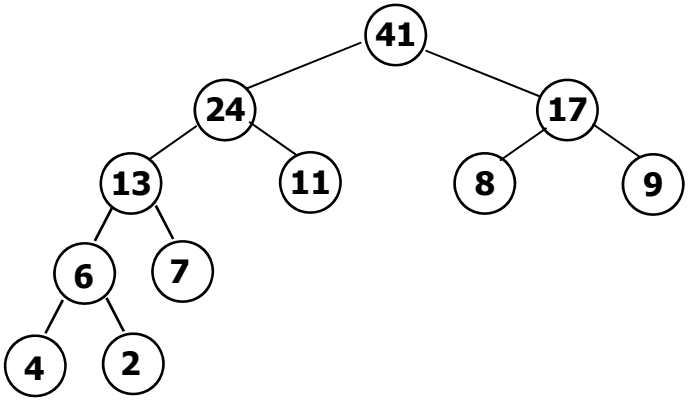
重点掌握的内容是①基本概念；②数据元素存储结构；③算法效率估算方法。

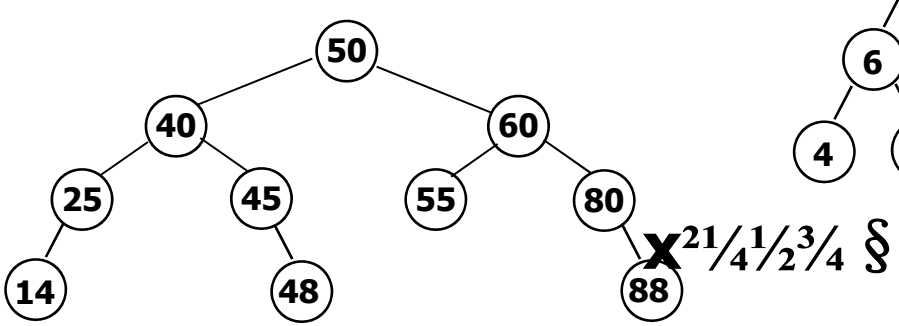


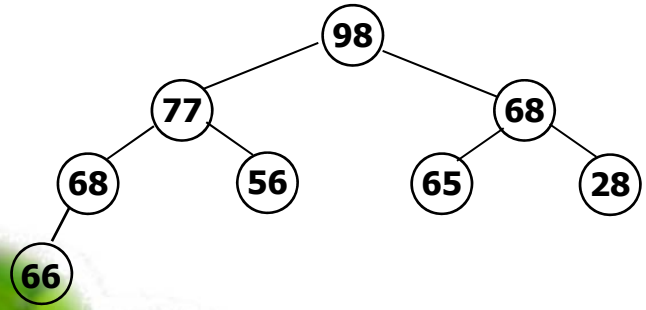
04 13 27 38 49 49 55 65 76 99

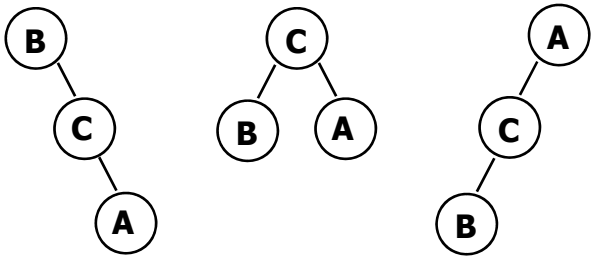
49 38 65 97 76 13 27 49 55 04


50
50 50
50 50 50 50
50 50 50 50 50 50 50 50

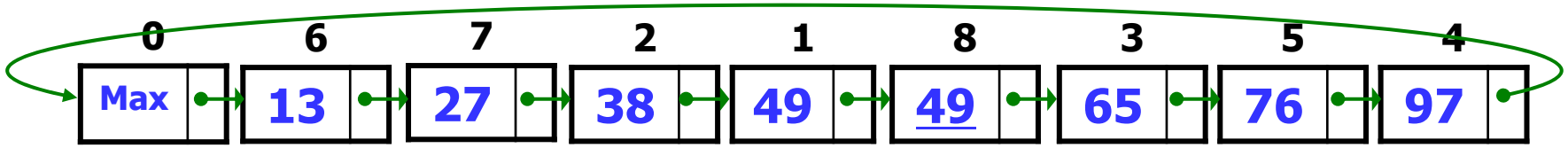

41
24 17
13 11 8 9
6 7
4 2

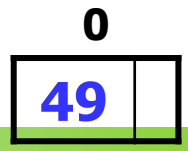

50
40 60
25 45 55 80
14 48 88


98
77 68
68 56 65 28
66

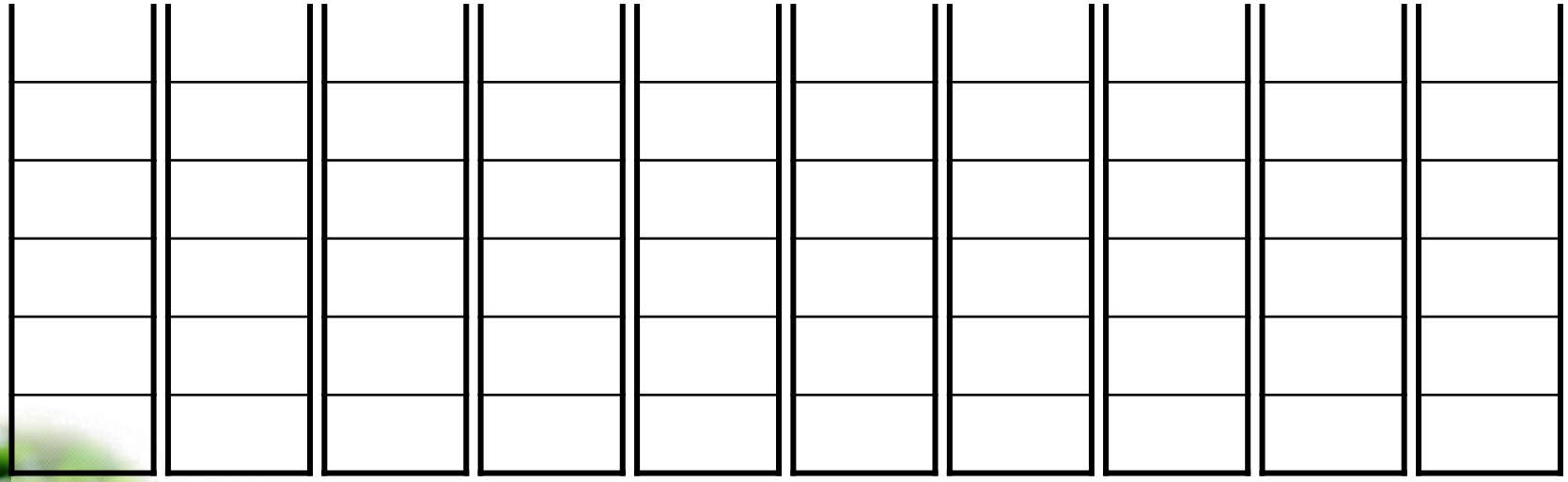

B
C
A
C
B A
A
C
B


38 38 65 97 76 13 27 49 97


0
6
7
2
1
8
3
5
4
Max
13
27
38
49
49
65
76
97


0
49

278 109 063 930 589 184 505 269 008 083



0	1	2	3	4	5	6	7	8	9
278	109	063	930	589	184	505	269	008	083

97 76 65 49 49 13 27 38