

数据结构

1 绪论

董洪伟 陈聪 周世兵

联系电话: 13812529213

E-mail: worldguard@163.com

主要内容

- 什么是数据结构
 - 定义、内容
- 基本术语
 - 数据：数据对象、数据元素、数据项
 - 数据结构：逻辑结构、物理结构
- 抽象数据类型
 - 定义、表示
- 算法和算法分析
 - 算法的概念、算法复杂度

什么是数据结构

- 程序 = 数据结构 + 算法

- Pascal之父, Niklaus Wirth
- 数据结构: 问题的数学模型 — 数据表示
- 算法: 处理问题的策略 — 数据处理
- 程序: 一组指令 — 数据结构和算法的实现

- 计算机解决问题的步骤:

- (抽象出)数学模型
- (设计)求解算法
- (编制)程序

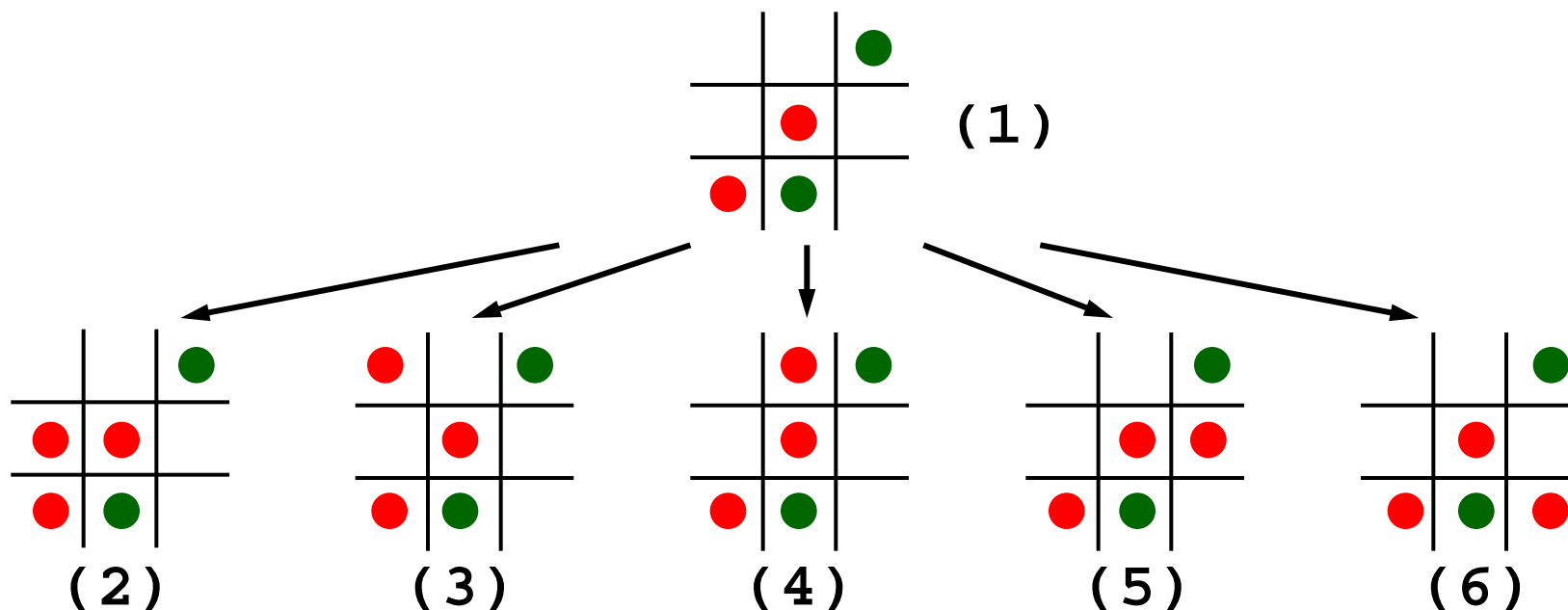
什么是数据结构

- 例：学生名单

学号	姓名	性别	籍贯	年龄
98131	张三	男	北京	20
98164	李斯	女	上海	21
98165	王武	男	广州	19
98182	赵柳	女	香港	22
98224	...			

什么是数据结构

• 例：井字棋人机对弈问题



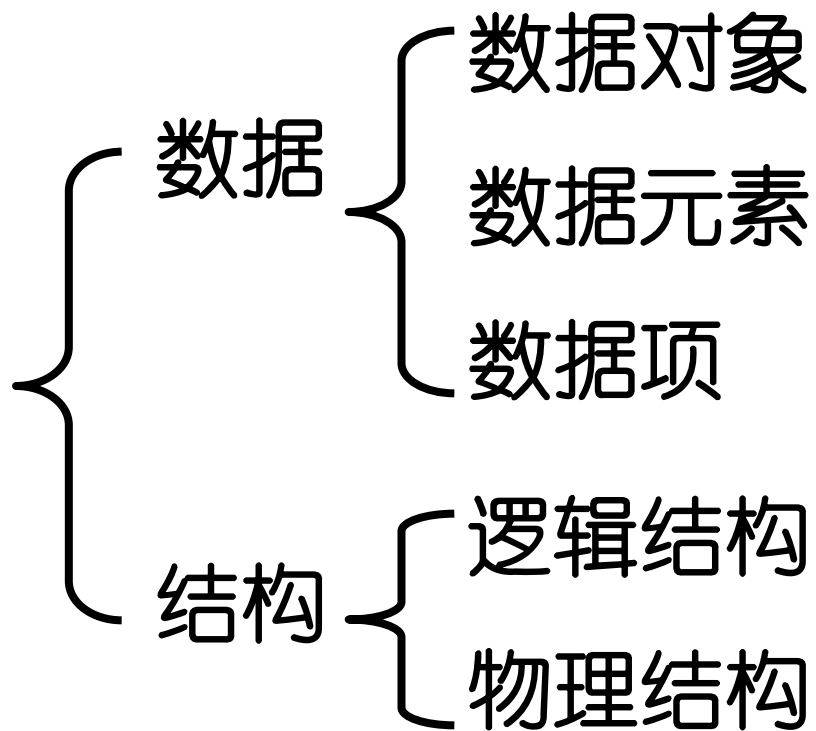
什么是数据结构

• 例：城市交通咨询图



查询两地的最短路径

基本术语



基本术语：数据

- **数据：对客观事物的符号表示**
 - 指所有需输入计算机并被程序所处理的对象的总称（在计算机科学中）
- **例如：图书借阅管理系统**
 - 图书信息：

书号	书名	借阅者编号
001	理论力学	9002
002	高等数学	9001
...


基本术语：数据元素

- 数据元素

— 数据的基本单位，在程序中常作为一个整体考虑和处理

书号	书名	借阅者编号
001	理论力学	9002
002	高等数学	9001
...

一个数据元素
(一条记录)



基本术语：数据项

- 数据项

- 数据的不可分割的最小单位
- 一个数据元素可由一个或多个数据项构成

3个数据项



书号	书名	借阅者编号
001	理论力学	9002
002	高等数学	9001
...

基本术语：数据对象

- 数据对象

- 性质相同的数据元素的集合
- 是数据(全集)的子集

教科书 {	书号	书名	借阅者编号
	001	理论力学	9002
	002	高等数学	9001

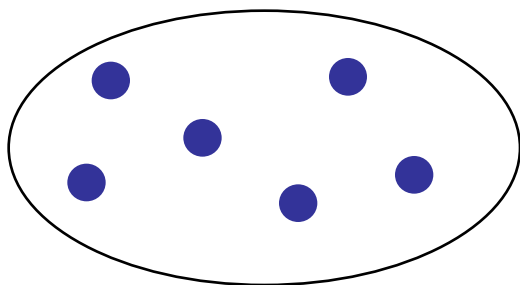
基本术语：逻辑结构

- 逻辑结构

- 数据元素之间的逻辑关系
- 即在现实世界中的关系

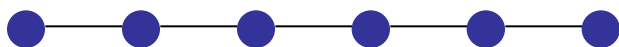
- 分类

- 集合：同在一个集合中
 - 比如同班同学之间

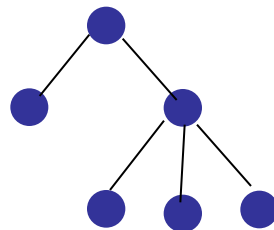


基本术语：逻辑结构

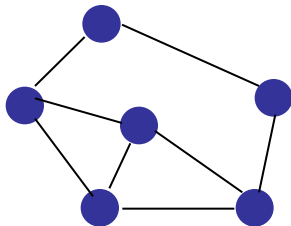
- 线性结构：一个接一个
 - 比如学生花名册中的记录之间



- 树形结构：一对多
 - 比如家谱



- 图形结构：多对多
 - 比如地图



基本术语：物理结构

- 物理结构

- 即存储结构
- 数据元素在计算机中存储时的关系

- 顺序映像（顺序存储结构）

- 以存储地址上的联系来体现数据元素间的逻辑关系

- 非顺序映像（链式存储结构）

- 通过指针的指向来体现数据元素间的逻辑关系

基本术语：物理结构

• 例：学生花名册

学号	姓名	性别	籍贯	年龄
98131	张三	男	北京	20
98164	李斯	女	上海	21
98165	王武	男	广州	19
98182	赵柳	女	香港	22
98224	...			

– 数据元素之间的逻辑关系：线性关系

基本术语：物理结构

- 顺序存储：各个数据元素在计算机中的存储地址也是线性关系，即连续存放

98131
张三
男
98164
李斯
女
.....



一个数据元素

```
typedef struct{
    string id,name;
    bool boy;
}student;
student students[60];
```


基本术语：物理结构

优点：随机存取

缺点：

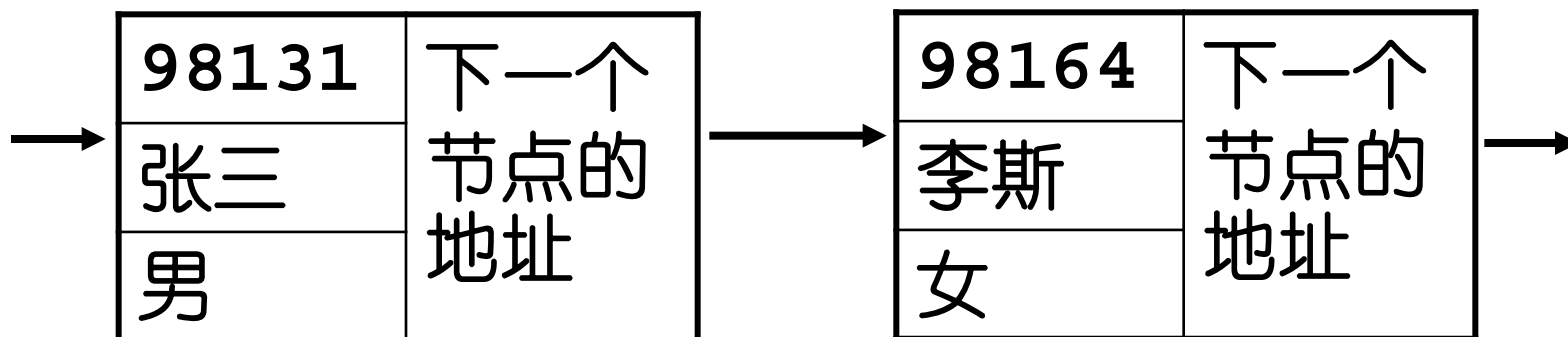
1、数组满时，无法继续插入学生成绩，设置较大的数组空间，又可能会浪费存储空间。

解决办法：动态数组
(`malloc`, `realloc`, `free`)

2、插入、删除元素要大量移动记录

基本术语：物理结构

- 链式存储：各个数据元素在计算机中的存储位置任意，通过指针相互连接起来



```
typedef struct{  
    string id,name;  
    bool boy;  
}student;
```

```
struct LNode {  
    student stu;  
    struct LNode *next;  
};  
struct LNode *p;
```

基本术语：物理结构

优点：

插入、删除方便，且只要内存空间够大，就不会满。

缺点：不能随机存取

抽象数据类型

- 抽象数据类型

- Abstract Data Type

- 一个数学模型及定义在该模型上的一组操作，用三元组 (D, S, P) 表示：

- D ：数据对象

- S ： D 上关系的集合

- P ： D 上基本操作的集合

抽象数据类型

- 格式

```
ADT  抽象数据类型名{  
    数据对象： <数据对象的定义>  
    数据关系： <数据关系的定义>  
    基本操作： 基本操作名(参数表)  
                初始条件： <初始条件描述>  
                操作结果： <操作结果描述>  
}ADT  抽象数据类型名
```

- 例子： 课本P9

算法和算法分析：概念

- **算法 (Algorithm)**

- 算法是对问题求解步骤的描述
- 是指令的有限序列，其中每条指令表示一个或多个操作

算法和算法分析：概念

• 算法的特性

– 一个正确的算法必须满足

- 有穷性：算法在执行有穷步后结束，且每步可在有穷时间内完成
- 确定性：每个步骤都有确切的含义，相同的输入具有相同的执行路径和结果
- 可行性：算法中各操作可通过已实现的基本运算执行有限次完成
- 输入：零或多个输入
- 输出：一或多个输出

算法和算法分析：概念

- 算法设计的要求

- 一个好的算法应当满足：

- 正确性：算法应能满足具体问题的需求
 - 可读性：算法应易于阅读和理解
 - 健壮性：输入数据非法时，算法也能适当作出反应或进行处理
 - 高效性：算法执行时间短，占用存储空间少

算法和算法分析：时间复杂度

- 算法时间效率的量度

- 事后统计法

- 测量一个算法执行所需要的时间

- 代码：C++教材P25

- 缺点：

- 需要编写测试程序

- 测量结果依赖于具体的软、硬件

- 事前分析估算法

算法和算法分析：时间复杂度

- 事前分析估算法

- 一个程序的执行时间取决于如下因素：

- 算法
 - 问题的规模
 - 编程语言
 - 编译程序
 - 硬件性能

算法和算法分析：时间复杂度

- 其中：

- 同一个算法在不同的语言、编译程序和硬件的条件下，执行时间是不同的
- 所以评价算法的优劣应当排除这三者的影响
- 比如：算法A在硬件A上执行时间为1秒，算法B在硬件B上执行时间为2秒，并不能因此就认为算法A的效率更高
- 因此只需要考虑算法本身和问题的规模

算法和算法分析：时间复杂度

• 渐进时间复杂度

```
for(i=1; i<=n; ++i)
    for(j=1; j<=n; ++j) {
        c[i][j]=0;
        for(k=1; k<=n; ++k)
            c[i][j]+=a[i][k]*b[k][j];
    }
```

- 该程序最基本的操作是 $a[i][k]*b[k][j]$
- 其执行次数 $F = n^3$
- 整个程序的执行时间和 F 成正比

算法和算法分析：时间复杂度

- 渐进时间复杂度

- 在算法中选择一种不可再分解的基本操作
- 该操作的重复次数应与算法的执行时间成正比，一般为问题规模 n 的函数 $f(n)$
- 此时可记算法的时间量度为：
 $T(n) = O(f(n))$

算法和算法分析：时间复杂度

• 例1

```
for(i=2; i<=n; ++i)
    for(j=2; j<=i-1; ++j) {
        ++x;
        a[i][j] = x;
    }
```

i=2: 0次
i=3: 1次
.....
i=n: n-2次

– 基本操作++x的执行次数

$$= \frac{(n-1)(n-2)}{2} = \frac{n^2 - 3n + 2}{2}$$

– 因此渐进复杂度= $O(n^2)$

• 例2

```
for(i = 1; i <= n; i++) {  
    m = i;  
    for(j = i; j <= n; j++)  
    {  
        if(data[j] < data[m])  
            m = j;  
  
        if(m != i)  
        {  
            temp    = data[m];  
            data[m] = data[i];  
            data[i] = temp;  
        }  
    }  
}
```

执行次数= n ,
 $T(n)=O(n)$

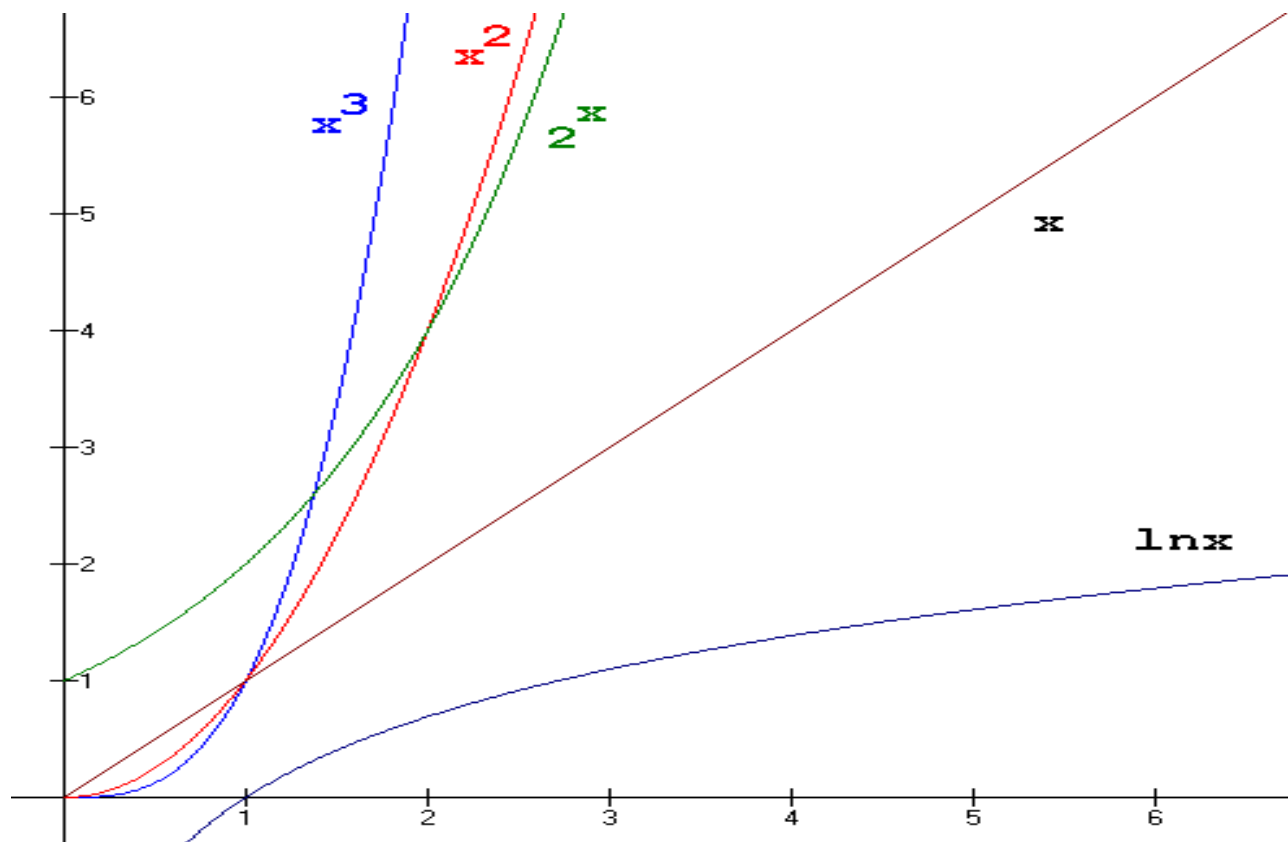
执行次数都是
 $n(n+1)/2$,
 $T(n)=O(n^2)$

– 对于整个算法, $T(n)=O(n^2+n)=O(n^2)$

算法和算法分析：时间复杂度

- 各种常见的渐进复杂度

– $a^n > n^b > \log_c n$



算法和算法分析：空间复杂度

- 空间复杂度

- 空间复杂度是算法执行时所需存储空间的量度，记作： $S(n)=O(f(n))$ ，其中 n 为问题的规模
- 一般不考虑存放数据本身占用的空间，只考虑执行算法所需辅助空间，除非数据所占空间与算法本身有关

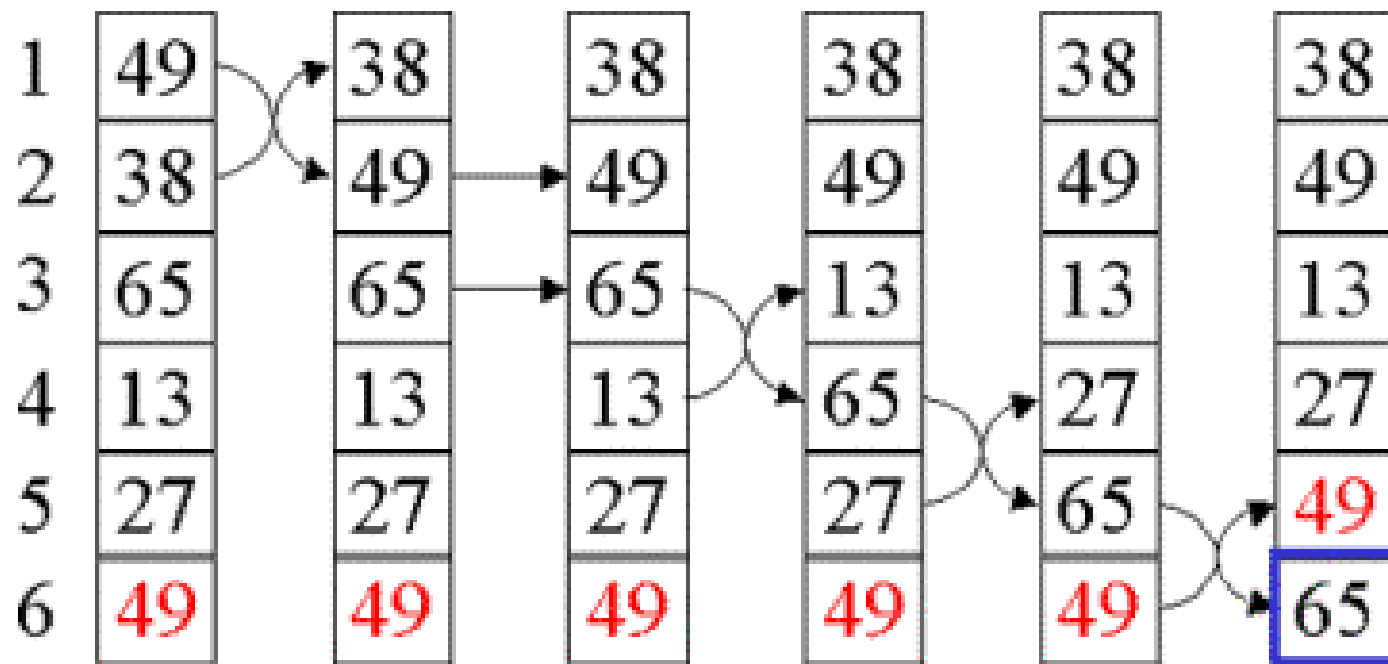
算法和算法分析

- 有些算法的复杂度与输入数据有关
 - 比如气泡排序，当输入数据基本有序时，其时间复杂度为 $O(n)$ ，基本无序时，为 $O(n^2)$ ，平均为 $O(n^2)$
 - 此时就应该分最好情况、最差情况、平均情况来讨论

本章小结

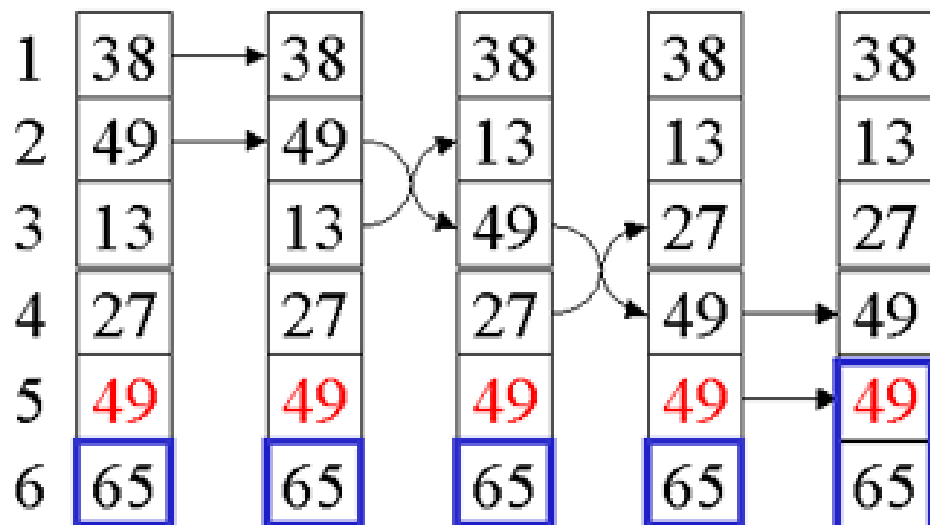
- 基本概念
 - 了解
- 算法复杂度的量度
 - 掌握量度和表示的方法

附录：冒泡排序示意图

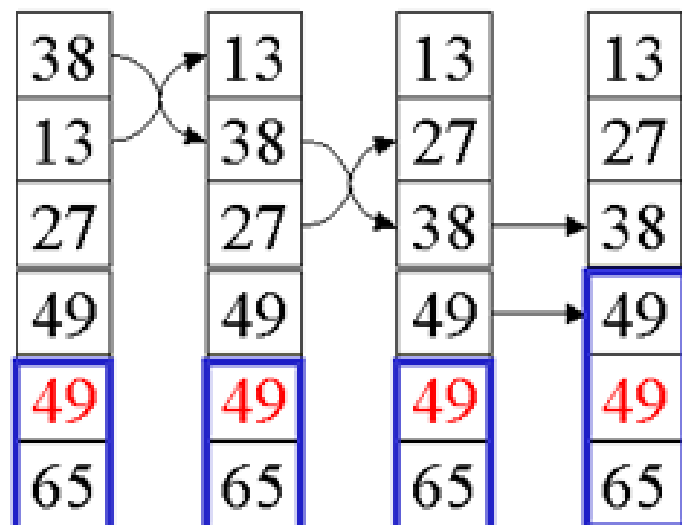


附录：冒泡排序示意图

第二趟：



第三趟：



附录：主讲教材

- 《数据结构》（C语言版） 严蔚敏等编著。
- 数据结构（用面向对象方法和C++描述） 殷人昆等编著。清华大学出版社

附录：实验教材

- 《数据结构题集》（C语言版） 严蔚敏等编著。清华大学出版社