数据结构
Data Structures

张海军

教授、博士生导师

大数据技术研究中心

哈工大企业与服务计算研究中心（ICES）

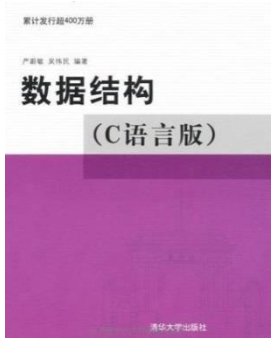
深圳市互联网信息协同技术及应用重点实验室

课程安排

- 课程编号：COMP2022
- 授课学时：40（5-15周，4学时/周）
- 实验学时：16（3/9、3/10、3/12、4/14、3/16周）
- 课程分类：专业（技术）基础
- 考核形式：
  - 期末笔试70% + 实验20% + 平时作业和测验10%
- 主讲教师：张海军
  - 联系方式：G栋710室/电话26033086（O）
  - Email：hjzhang@hit.edu.cn
- 助课教师：夏文
- 助教：于琼、杨沃佳、伍映吉、任狼、谢鹏飞、姬喜洋、黄中元、曲子奇

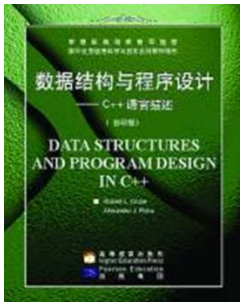
教材选择

- 课程资源：
  - 课件PPT：集成了各个教材内容，可作为简要学习内容。
  - 主要教材：
    - 《数据结构（C语言版）》
      - 严蔚敏、吴伟民 编著，清华大学出版社



  - 原版教材：
    - Data Structures and Program Design in C++

      数据结构与程序设计——C++语言描述（影印版）
    - Robert L. Kurse, Alexandeer J. Ryba



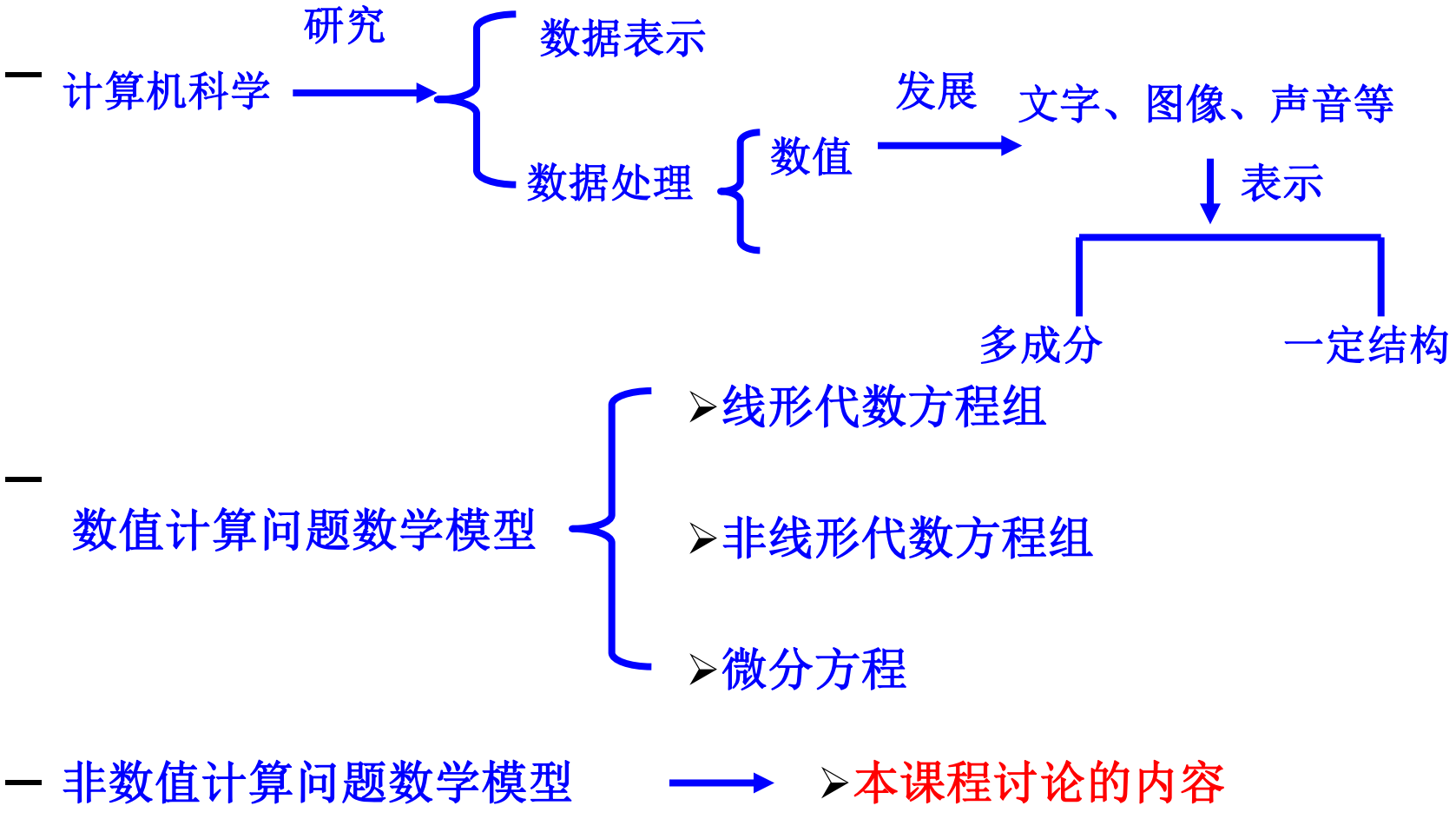
  - 参考书目：
    - 数据结构与算法（第4版）
      - 廖明宏、郭福顺、张岩、李秀坤，高等教育出版社，2007.
  - INTERNET

课程地位

- “数据结构”
  - 是计算机专业的核心课程之一，也是其他非计算机专业的主要选修课程之一。
- “数据结构”在计算机科学中是一门综合性的专业基础课。
- “数据结构”
  - 是计算机硬件、操作系统、编译原理、计算机网络、数据库系统及其他系统程序和大型应用程序的重要基础。
- “数据结构”正在不断发展。例如：多维图像数据结构等。
- 面向各专门领域中特殊问题的数据结构的研究和发展。

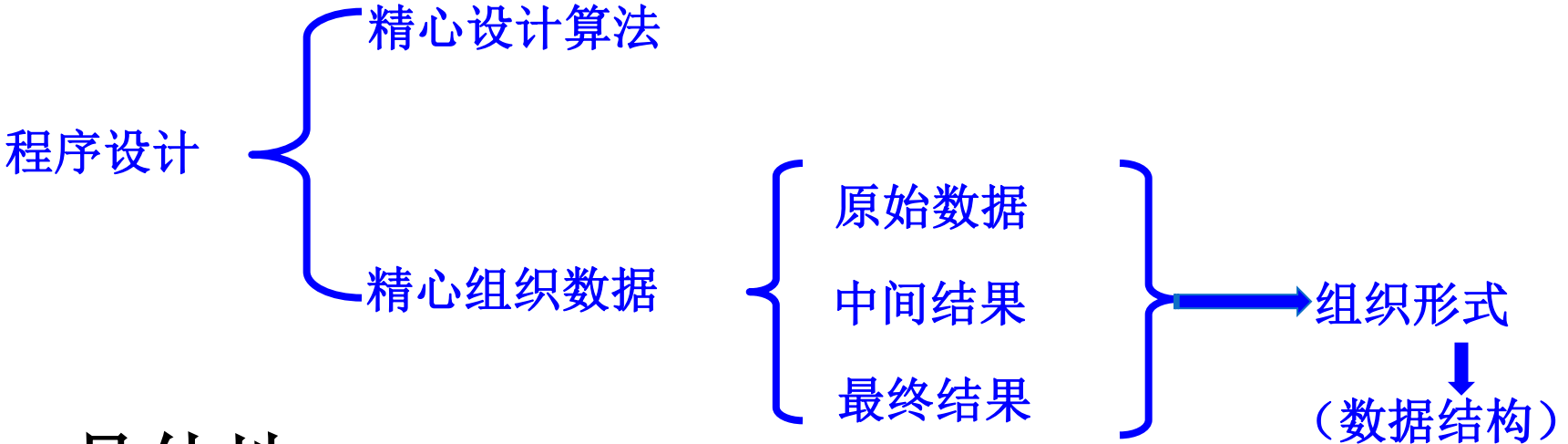
教学内容

- 课程来源：

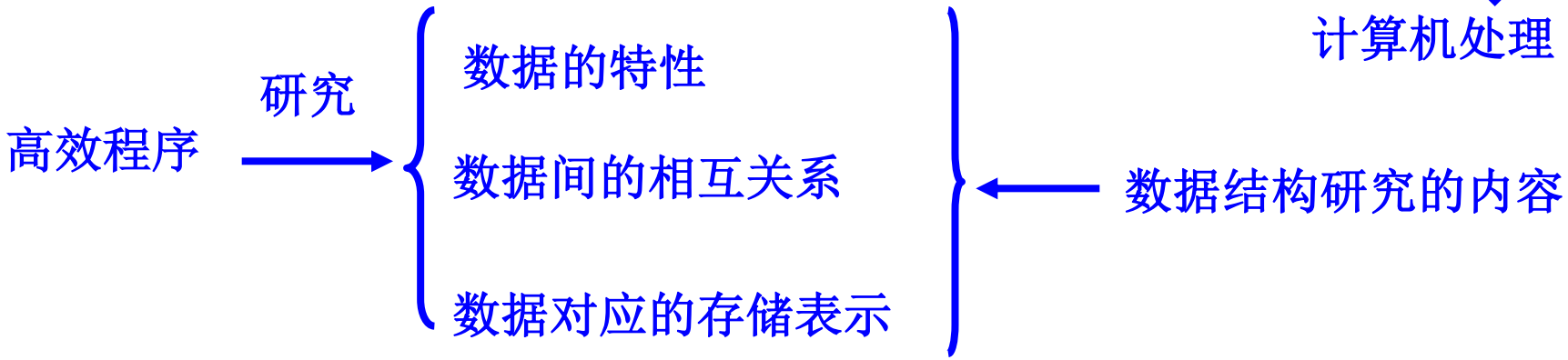

计算机科学
研究
数据表示
数据处理
数值
发展
文字、图像、声音等
表示
多成分
一定结构
数值计算问题数学模型
线形代数方程组
非线形代数方程组
微分方程
非数值计算问题数学模型
本课程讨论的内容

教学内容

- 与程序设计的关系：

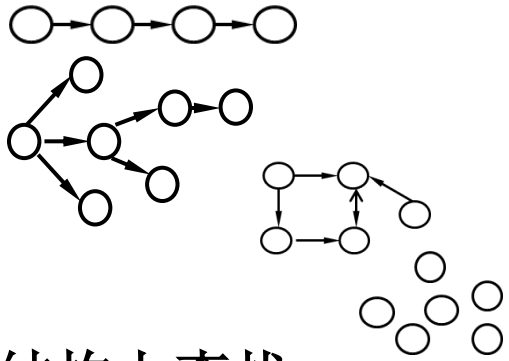


- 具体地：



课程一览

- 两个主题：
  - 数据结构：数据及其之间的相互关系，是对数据的抽象。
  - 算法：是求解特定问题的方法，是将输入转为输出的一系列计算步骤。
- 课程的内容组织：
  - 四种基本的数据结构
    - 线性结构：一对一的线性关系；
    - 树型结构：一对多的层次关系；
    - 图型结构：多对多的任意关系；
    - 集合结构：属于同一个集合关系。



  - 查找：线性结构、树型结构和散列结构上查找；
  - 排序：内部排序和外部排序（包括文件）；
  - 高级主题：分布式数据结构及其应用。

学习目标

- 让“数据结构”成为你编程的利器
  - 彻底精通数据结构。
  - 不仅仅停留在逻辑层面。
- 让“数据结构”
  - 不再是你程序设计的障碍！
  - 不再是你学习其他课程的障碍！
  - 不再是你 考研的障碍！
  - 不再是你找工作的障碍！
- 在编程中升华你的数据思维，爱上计算机专业！

第1章 绪论

本章重点与难点

1. 掌握数据结构等基本概念。

2. 数据的逻辑结构、存储结构以及两者之间的关系；

3. 算法及特性；

4. 掌握估算算法时间复杂度的方法。

第一章 绪论

1.1 数据结构及其讨论范畴

1.2 基本概念和术语

1.3 抽象数据类型的表示与实现

1.4 算法和算法分析

第一章 绪论

1.1 数据结构及其讨论范畴

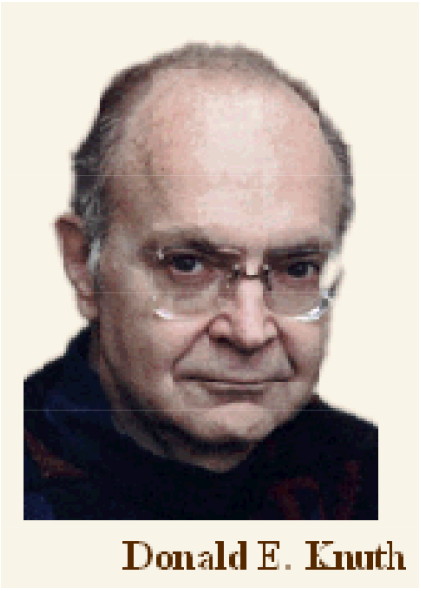
1.2 基本概念和术语

1.3 抽象数据类型的表示与实现

1.4 算法和算法分析

1.1 数据结构及其讨论范畴

- 数据结构的兴起与发展
  - 数据结构的创始人—Donald.E.Knuth



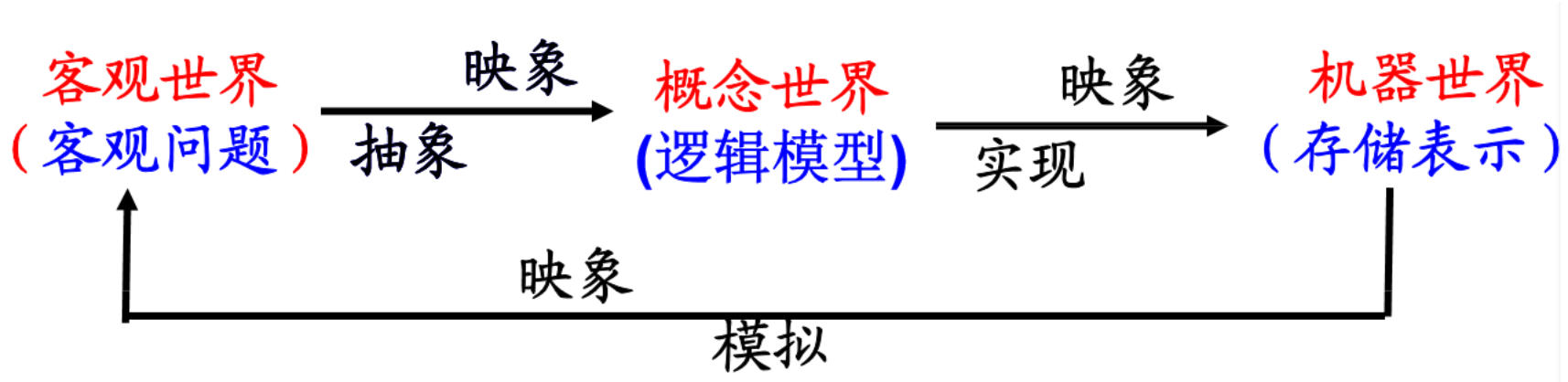
- 1938年出生，25岁毕业于加州理工学院 数学系博士毕业后留校任教，28岁任 副教授。30岁时，加盟斯坦福大学计算 机系，任教授。从31岁起，开始出版他 的历史性经典巨著：

  The Art of Computer Programming

- 他计划共写7卷，然而出版三卷之后， 已震惊世界，使他获得计算机科学界的最高荣誉图灵奖，此时，他年仅36岁。

1.1 数据结构及其讨论范畴

- 客观世界与计算机世界的关系
  - 计算机科学是研究信息表示和信息处理的科学。
  - 信息在计算机内是用数据表示的。
  - 用计算机解决实际问题的实质可以用下图表示：



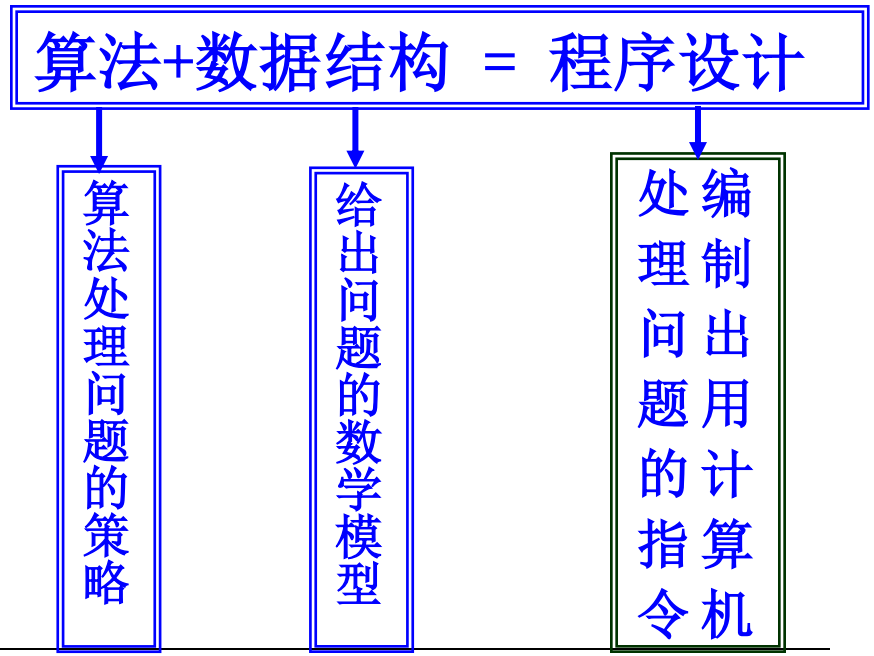
1.1 数据结构及其讨论范畴

- 从问题到程序

问题 ⟶ 解决问题的算法 ⟶ 实现算法的程序

- 美国著名计算机科学家、Pascal语言发明者Niklaus Wirth 在1976年出版的一本书的书名为：

“算法+数据结构=程序设计”

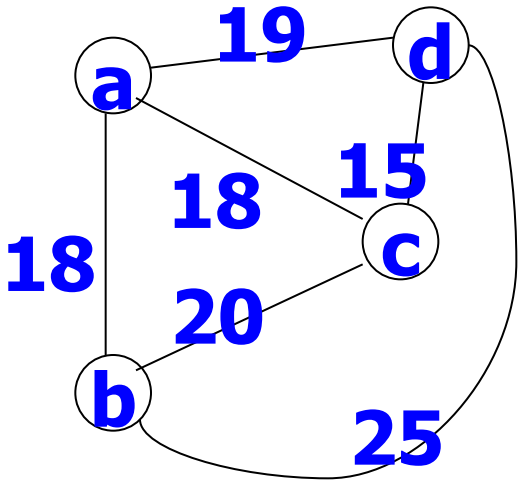


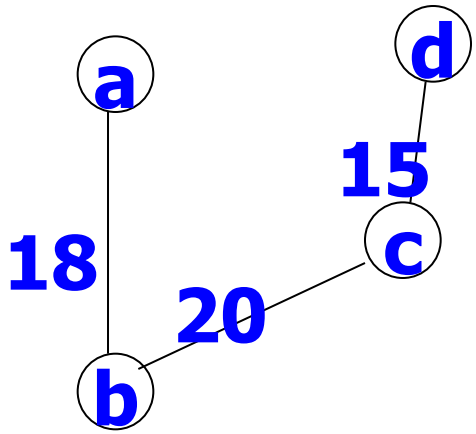
1.1 数据结构及其讨论范畴

- 程序设计的实质是什么
  - 对确定的问题选择一种好的结构，加上设计一种好的算法，需要解决两个问题：即数据结构（问题的数学模型）和算法（处理问题的策略）。
  - 数据表示：将数据存储在计算机中
  - 数据处理：处理数据，求解问题
  - 数据结构问题起源于程序设计
- 数据结构随着程序设计的发展而发展
  - ① 无结构阶段：在简单数据上作复杂运算
  - ② 结构化阶段：数据结构＋算法＝程序
  - ③ 面向对象阶段：（对象＋行为）＝程序
- 数据结构的发展并未终结。。。

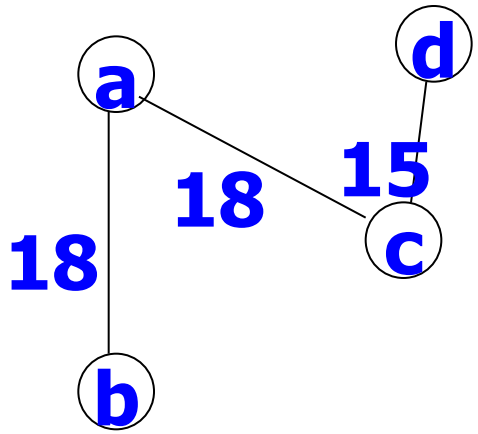
1.1 数据结构及其讨论范畴

- 例1 煤气管道的铺设问题。
  - 在城市的各小区之间铺设煤气管道，共有4个小区，由于地理环境不同等因素使各条管线所需投资不同，如何使投资成本最低？





53



51

1.1 数据结构及其讨论范畴

- 例2 求n个整数中的最大值。
- 例3 交叉路口的红绿灯管理问题。
- 例4 ……

说明：

例子中的数学模型正是数据结构要讨论的问题。

1.1 数据结构及其讨论范畴

例 学籍管理问题（表结构）

– 已知某级学生情况，要求分班按入学成绩排列顺序。

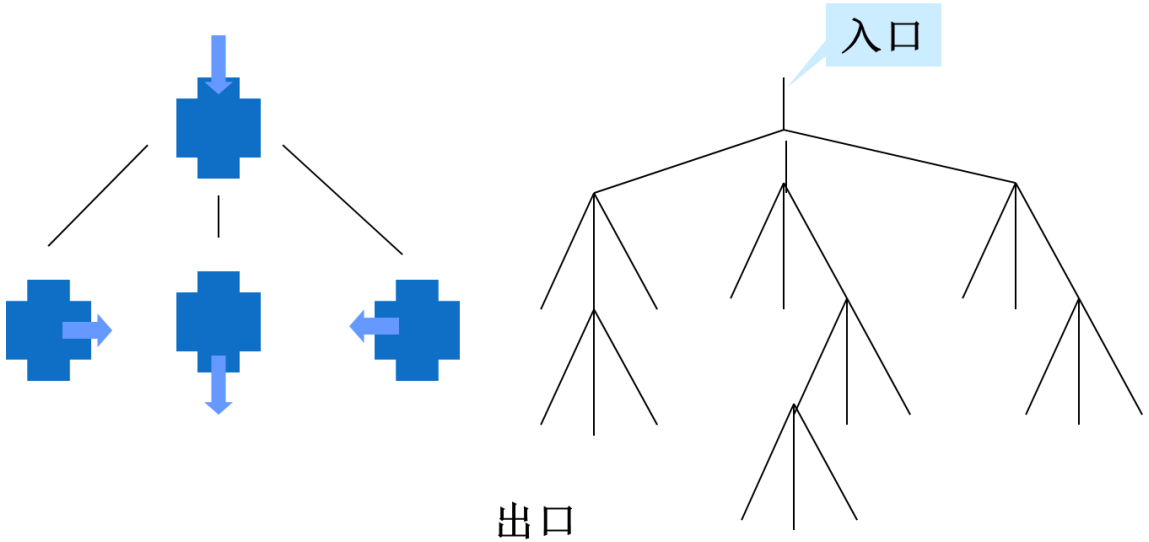
学号	姓名	性别	出生日期	籍贯	入学成绩	所在班级
00201	杨润生	男	82/06/01	广州	561	00计算机2
00102	石磊	男	83/12/21	汕头	512	00计算机1
00202	李梅	女	83/02/23	阳江	532	00计算机2
00301	马耀先	男	82/07/12	广州	509	00计算机3

说明：

在此类文档管理的数学模型中，计算机处理的对象之间通常存在着一种最简单的线性关系，该数学模型称为线性模型。

1.1 数据结构及其讨论范畴

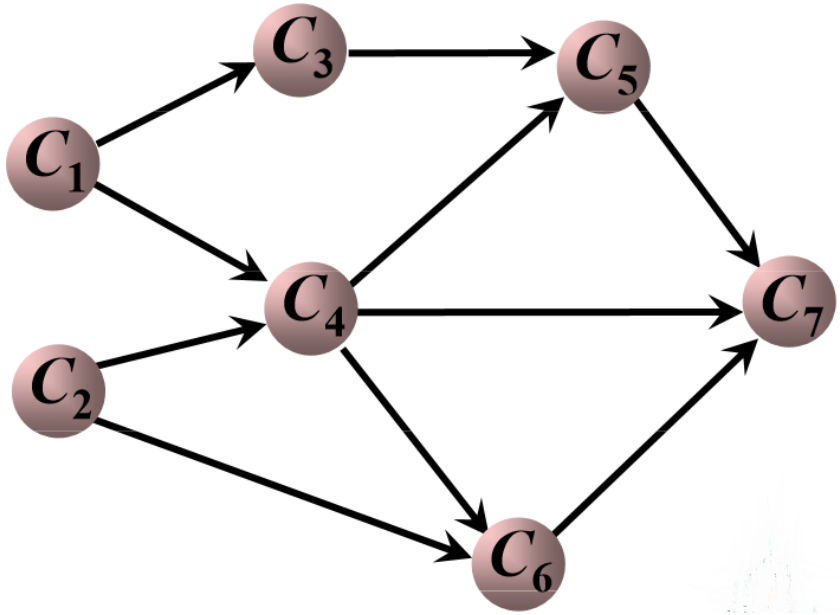
- 例 迷宫问题（树结构）
  - 在迷宫中，每走到一处，接下来可走的通路有三条。计算机处理的这类对象之间通常不存在线性关系。若把从迷宫入口处到出口的过程中所有可能的通路都画出，则可得一棵“树”。



1.1 数据结构及其讨论范畴

- 例 教学计划编排问题（图结构）
  - 如何表示课程之间的先修关系？

编号	课程名称	先修课
C_1	高等数学	无
C_2	计算机导论	无
C_3	离散数学	C_1
C_4	程序设计	C_1, C_2
C_5	数据结构	C_3, C_4
C_6	计算机原理	C_2, C_4
C_7	数据库原理	C_4, C_5, C_6



1.1 数据结构及其讨论范畴

- 计算机求解问题
  - 问题→抽象出问题的模型→求模型的解
  - 问题——数值问题、非数值问题
    - 数值问题→数学方程
    - 非数值问题→数据结构
- 数据结构讨论的问题：

① 数据的逻辑结构

- 数据元素之间的逻辑关系，是具体关系的抽象。

② 数据的存储结构(物理结构)

- 数据元素及其关系在计算机内存中的表示；

③ 数据的运算

- 即对数据施加的操作。定义在数据的逻辑结构上的抽象的操作。

1.1 数据结构及其讨论范畴

- 数据结构是一门讨论"描述现实世界实体的数学模型(非数值计算)及其上的操作在计算机中如何表示和实现"的学科。

  - 在解决问题时可能遇到的典型的逻辑结构

    ----(数据逻辑结构)

  - 逻辑结构的存储映象----(存储实现)

  - 建立在二者之上的基本运算----(操作)

第一章 绪论

1.1 数据结构及其讨论范畴

1.2 基本概念和术语

1.3 抽象数据类型的表示与实现

1.4 算法和算法分析

1.2 基本概念和术语

- **数据**：一切能输入到计算机中并能被计算机程序识别和处理的符号集合。
  - 数值数据：整数、实数等
  - 非数值数据：图形、图象、声音、文字等

例 学生（数据元素）

姓名	性别	年龄	专业	班级

数 据 项

- **数据元素**：数据的基本单位，在计算机程序中通常作为一个整体进行考虑和处理。
- **数据项**：构成数据元素的不可分割的小单位。
- **数据对象**：具有相同性质的数据元素的集合。
- **结点**：数据元素在计算机内的位串表示 。
- **域（字段）**：数据元素中数据项在计算机内的表示 。
- **信息表**：是数据对象在计算机内的表示。

1.2 基本概念和术语

- 数据结构
  - 数据元素之间的相互关系，这种关系是抽象的，即并不涉及数据元素的具体内容。是数据元素及其相互间关系的 数学描述 。
  - 相互之间存在一定关系的数据元素的集合。
  - 按照视角的不同，数据结构分为逻辑结构和存储结构。建立在二者之上的基本运算。
- 数据的逻辑结构
  - 数据元素之间的抽象关系，数据元素之间逻辑关系的整体 。
  - 数据的逻辑结构是从具体问题抽象出来的数据模型。

1.2 基本概念和术语

- **数据的存储结构**
  - 又称物理结构，是数据及其逻辑结构在计算机中的表示。
  - 实质上是内存分配，以确定元素及元素之间关系的表示。
  - 在具体实现时，依赖于计算机语言。

1.2 基本概念和术语

- 数据结构的形式定义描述为：

数据结构是一个二元组

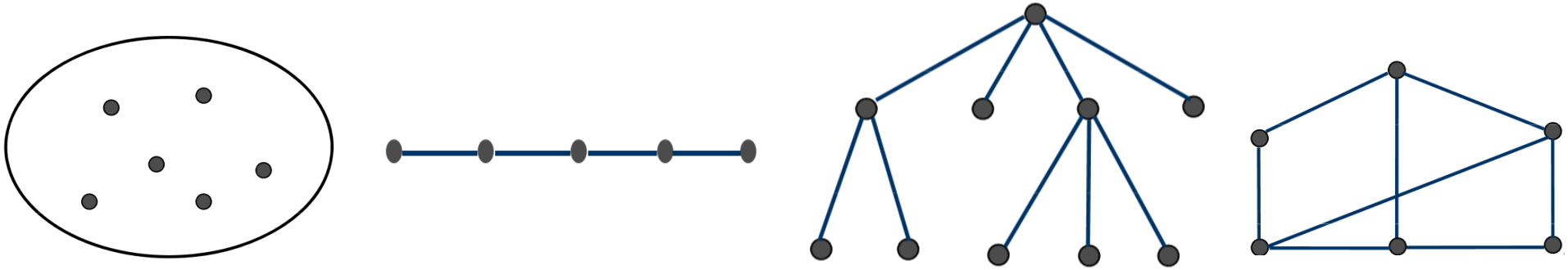
Data_Structures = (D, R)

其中：D 是**数据元素的有限集**，

R 是 D上**关系的有限集**。

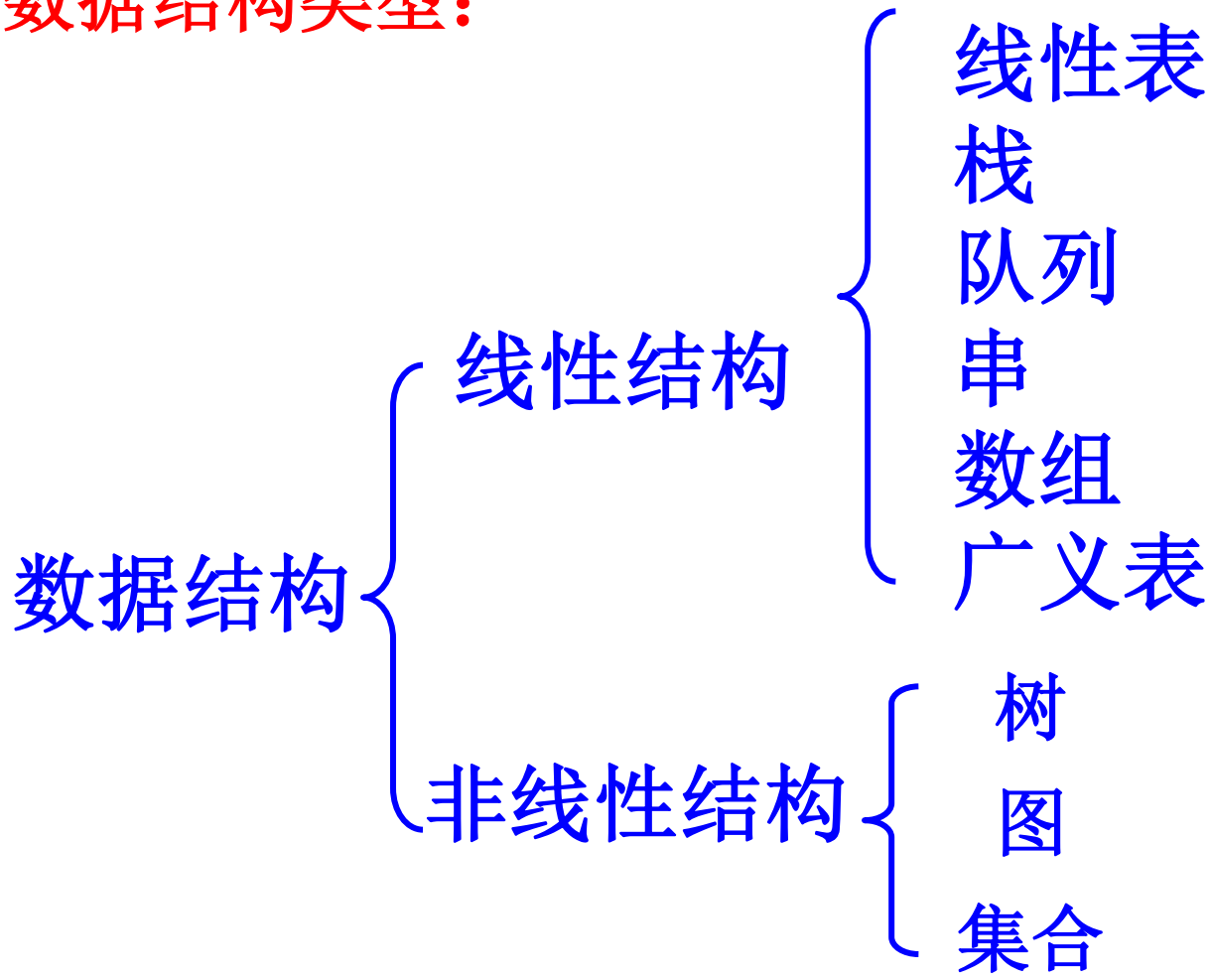
1.2 基本概念和术语

- 数据结构从逻辑上分为四类：
  即四种基本的逻辑结构
  - 集合：数据元素之间就是“属 于同一个集合”；
  - 线性结构：数据元素之间 存在着一对一的线性关系；
  - 树型结构：数据元素之间存在 着一对多的层次关系；
  - 图结构：数据元素之间存在着 多对多的任意关系。



1.2 基本概念和术语

- 数据结构类型：



1.2 基本概念和术语

- **数据的逻辑结构：**

（1）线性结构：除第一个元素和最后一个元素之外，其他元素都有且仅有一个直接前驱，有且仅有一个直接后继；

（2）非线性结构：其逻辑特征是一个结点可能有多个直接前驱和直接后继。

1.2 基本概念和术语

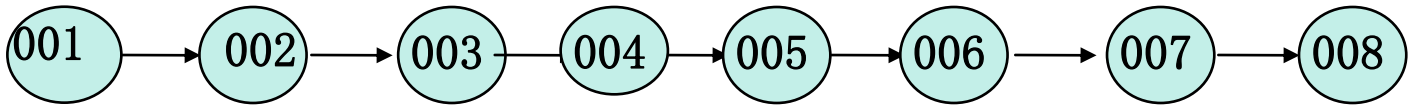
- **数据的逻辑结构：**

例

学生基本情况登记表，记录了每个学生的学号、姓名、专业、政治面貌、面貌，表中的记录是按学生的学号顺序排列的。

学号	姓名	专业	政治面貌
001	王洪	计算机	党员
002	孙文	计算机	团员
003	谢军	计算机	团员
004	李辉	计算机	团员
005	沈祥福	计算机	党员
006	余斌	计算机	团员
007	巩力	计算机	团员
008	孔令辉	计算机	团员

学生间学号顺序关系是一种线性结构关系



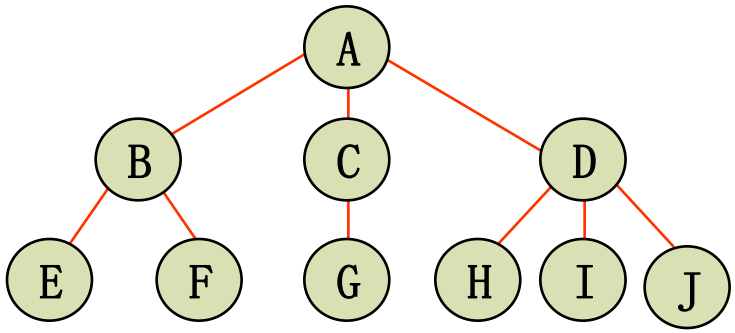
1.2 基本概念和术语

- **数据的逻辑结构：**

例

家族的族谱:假设某家族有10个成员A， B， C， D， E，F， G， H，I， J，他们之间的血缘关系可以用如下图表示。

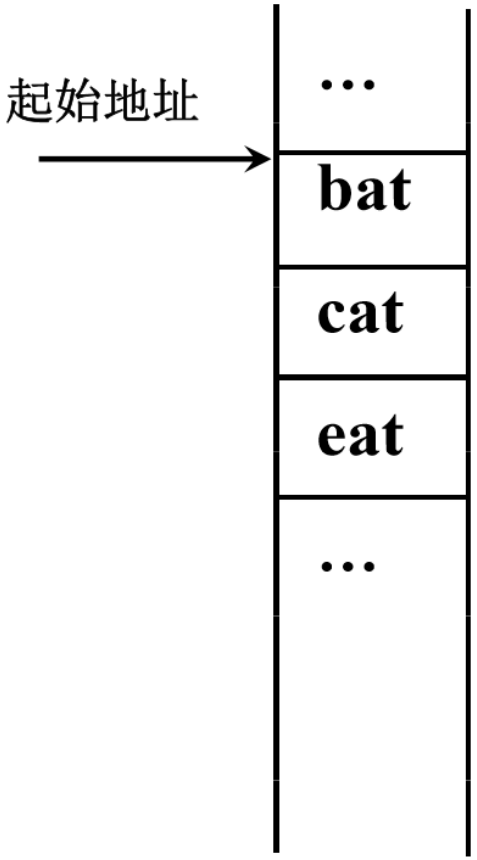
家族成员之间的关系
是一种非线性结构关系



1.2 基本概念和术语

- 两种基本的存储结构：
  - 顺序存储结构
    - 用一组连续的存储单元依次存储数据元素，数据元素之间的逻辑关系由元素的存储位置来表示。

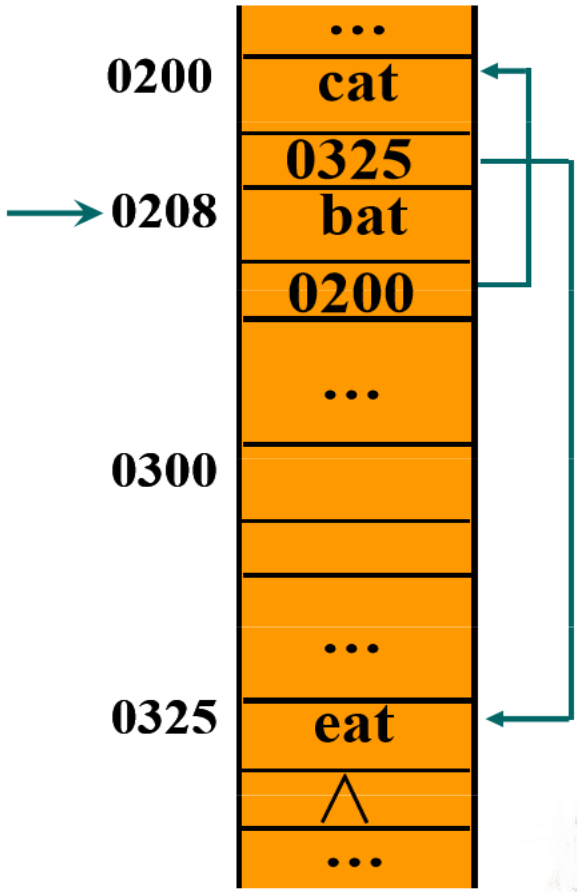
例：（bat, cat, eat）



1.2 基本概念和术语

- 两种基本的存储结构：
  - 顺序存储结构
    - 用一组连续的存储单元依次存储数据元素，数据元素之间的逻辑关系由元素的存储位置来表示。
  - 链接存储结构
    - 用一组任意的存储单元存储数据元素，数据元素之间的逻辑关系用指针来表示。

例：（bat, cat, eat）



1.2 基本概念和术语

- 数据的存储结构：
  - 顺序存储方法：数组
  - 链接存储方法：指针
  - 索引存储方法
  - 散列存储方法

说明：

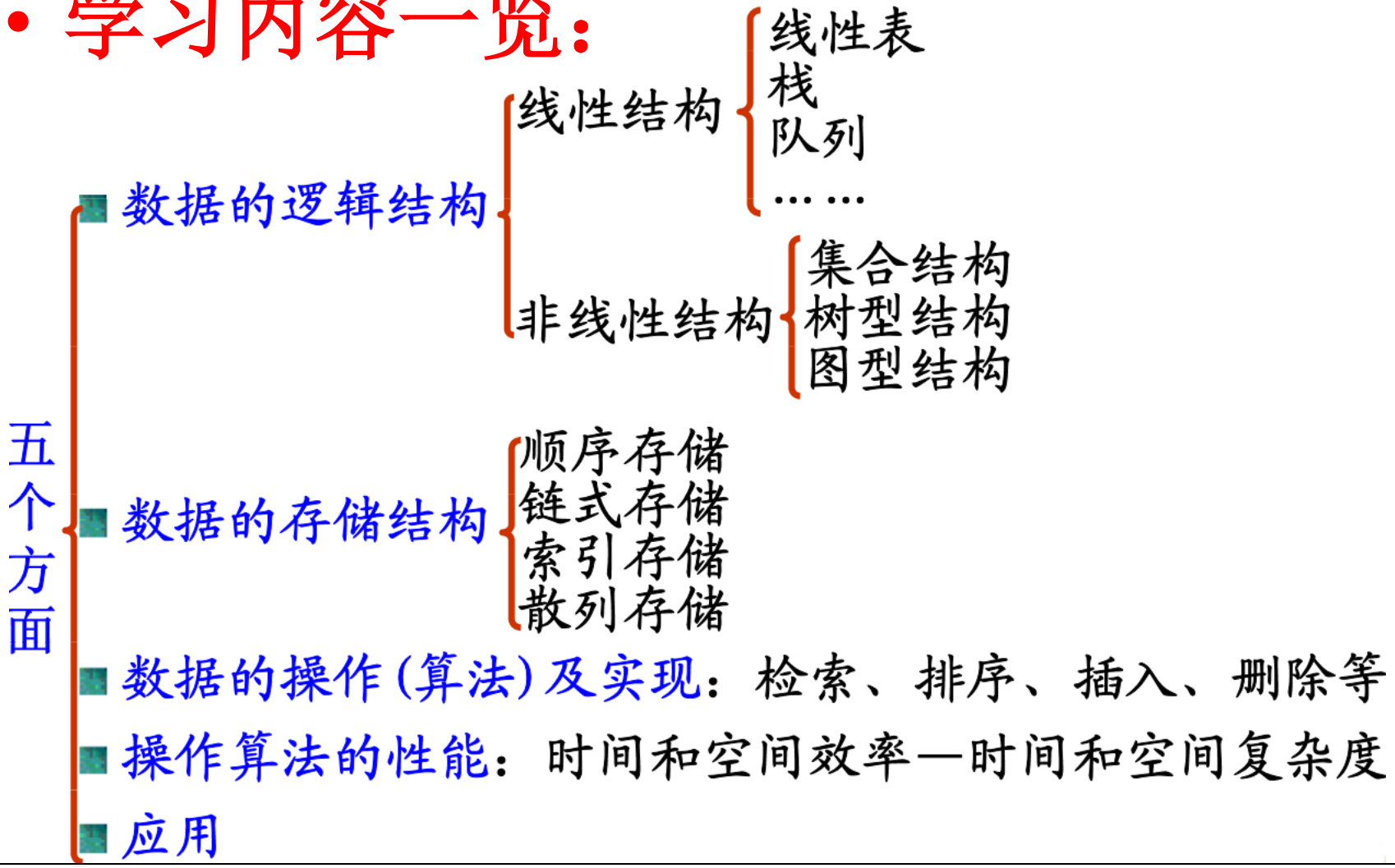
- 同一逻辑结构的不同存储结构，冠以不同的数据结构名称。如顺序表、链表、散列表。
- 运算不同，数据结构也不同。如栈和队列。更进一步，顺序栈、链栈、顺序队列、链队列。

1.2 基本概念和术语

- **逻辑结构与存储结构的关系：**
  - 数据的逻辑结构属于用户视图，是面向问题的，反映了数据内部的构成方式；数据的存储结构属于具体实现的视图，是面向计算机的。
  - 一种数据的逻辑结构可以用多种存储结构来存储，而采用不同的存储结构，其数据处理的效率往往是不同的。
- **数据结构的学习内容：**
  - 数据对象的结构形式，各种数据结构的性质(逻辑结构)；
  - 数据对象和“关系”在计算机中的表示(物理结构/存储结构)；
  - 数据结构上定义的基本操作(算法)及其实现；
  - 算法的效率（时间和空间）；
  - 数据结构的应用，如数据分类、检索等。

1.2 基本概念和术语

- 学习内容一览：



第一章 绪论

1.1 数据结构及其讨论范畴

1.2 基本概念和术语

1.3 抽象数据类型的表示与实现

1.4 算法和算法分析

1.3 抽象数据类型的表示与实现

- 抽象数据类型（Abstract Data Type）
  - 定义：一个数学模型和在该模型上定义的操作集合的总称 。
    - ADT是程序设计语言中数据类型概念的进一步推广和进 一步抽象。
    - 例：ADT int = ({x|x ∈Z}，{+,-,*,/,%, ≤,==})
    - 同一数学模型上定义不同的操作集，则它们代表不同的 ADT。
  - 实现：用适当的数据结构来表示ADT中的数学模型,并用一 组函数（方法）来实现该模型上的各种操作。

1.3 抽象数据类型的表示与实现

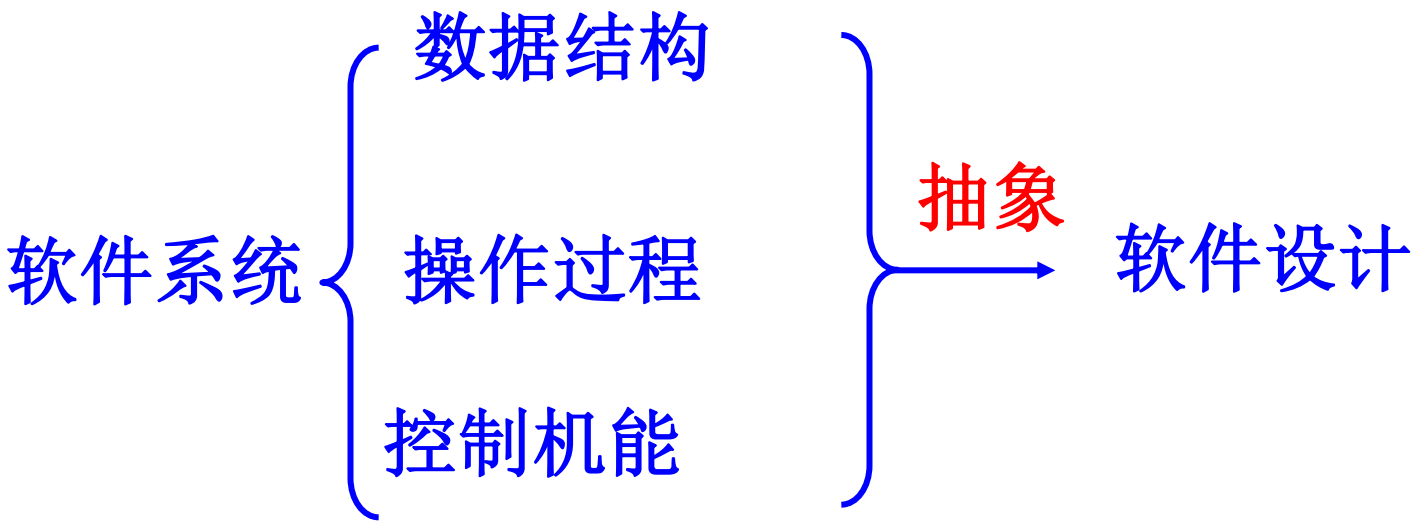
- **ADT特点：**
  - 不再局限于目前处理器中已经定义并实现的数据类型（固有数据类型），还包括用户在设计软件系统时自己定义的数据类型；
  - 近代程序设计方法学：软件系统的框架建立在数据之上，而不是操作之上；
  - 在构成软件系统的每个相对独立的模块上，定义一组数据和施于这些数据上的一组操作

在软件设计中，可以对哪三种不同的对象进行抽象？

数据类型的抽象层次越高，软件复用程度越高

1.3 抽象数据类型的表示与实现

- 软件设计中的对象抽象：



软件设计是对数据抽象、过程抽象和控制抽象。

1.3 抽象数据类型的表示与实现

- 抽象数据类型的形式描述

ADT = (D, S, P),其中：

D 是数据对象; S是 D 上的关系集;P是 D 的基本操作集。

- 抽象数据型的规格描述
  - 完整性：反映所定义的抽象数据型的全部特征；
  - 统一性：前后协调，不自相矛盾；
  - 通用性：适用于尽量广泛的对象；
  - 不依赖性：不依赖于程序设计语言。
- 规格描述的两个方面：语法和语义
  - 语法：给出操作的名称、I/O参数的数目和类型；
  - 语义：由一组等式组成，定义各种操作的功能及相互之间的关系。

1.3 抽象数据类型的表示与实现

- 抽象数据类型的作用
  - 抽象数据类型可以使我们更容易描述现实世界。例：用线性表描述学生成绩表，用树或图描述遗传关系。
- 抽象数据型的定义
  - 一个数学模型以及定义在该模型上的一组操作。
- 关键
  - 使用它的人可以只关心它的逻辑特征，不需要了解它的存储方式；
  - 定义它的人同样不必要关心它如何存储。
- 例子
  - 线性表这样的抽象数据类型，其数学模型是：数据元素的集合；该集合内的元素有这样的关系：除第一个和最后一个外，每个元素有唯一的前趋和唯一的后继；可以有这样一些操作：插入一个元素、删除一个元素等。

1.3 抽象数据类型的表示与实现

- 数据类型、数据结构和ADT
  - 各自含义：
    - 数据类型是一组值的集合
    - 数据结构则是数据元素之间的抽象关系;
    - 抽象数据型是一个数学模型及在该模型上定义的操作集的总称
  - 相互关系
    - 数据类型是根据数据结构分类的,同类型的数据元素的数据 结构相同。
    - 数据结构则是抽象数据型中数学模型的表示;
    - ADT是数据类型的进一步推广和进一步抽象。

第一章 绪论

1.1 数据结构及其讨论范畴

1.2 基本概念和术语

1.3 抽象数据类型的表示与实现

1.4 算法和算法分析

1.4 算法和算法分析

- 算法（algorithm）的概念
  - 算法是对问题求解过程的一种描述，是为解决一个或一类问题给出的一个确定的、有限长的操作序列。
  - 算法描述
    - 自然语言；
    - 程序设计语言；
    - 类语言*；
    - 关于本书采用的类语言描述：C 或 C++

1.4 算法和算法分析

例 欧几里德算法—辗转相除法求两个自然数m和n的最大公约数。

- 自然语言描述
  - ①输入m和n；
  - ②求m除以n的余数r；
  - ③若 r等于0，则n为最大公约数，算法结束；否则执行第④步；
  - ④将n的值放在m中，将r的值放在n中；
  - ⑤重新执行第②步 。
- 优点：容易理解
- 缺点：冗长、二义性

1.4 算法和算法分析

例 欧几里德算法—辗转相除法求两个自然数m和n的最大公约数。

– 伪代码描述

1. r = m % n;
2. 循环直到r等于0
   - 2.1 m = n;
   - 2.2 n = r ;
   - 2.3 r = m % n;
3. 输出n;

– 表达能力强，抽象性强，容易理解

1.4 算法和算法分析

- 算法的五个基本特征：
  - 有穷性：算法中的操作步骤为有限个，且每个步骤都能在有限时间内完成。
  - 确定性：组成算法的操作必须清晰无二义性。
  - 可行性：算法中的所有操作都必须足够基本，都可以通过已经实现的基本操作运算有限次实现之。
  - 输入：作为算法加工对象的量值，通常体现为算法中的一组变量。某些算法的字面上可以没有输入，实际上已被嵌入算法之中。
  - 输出：它是一组与输入有确定关系的量值，是算法进行信息加工后得到的结果，这种确定关系即为算法的功能。

1.4 算法和算法分析

例 考虑下列两段描述：

```
(1)
void exam1()
{
      n=2;
     while (n%2==0)
           n=n+2;
     printf("%d\n",n);
}
```

这两段描述均不能满足算法的特征，试问它们违反了哪些特征？

```
(2)
void exam2()
{
     y=0;
     x=5/y;
     printf( "%d,%d\n" ,x,y);
}
```

解：
(1)算法是一个死循环，违反了算法的有穷性特征。
(2)算法包含除零错误，违反了算法的可行性特征。

1.4 算法和算法分析

- 在设计算法时通常应考虑以下原则：
  - 算法必须是“正确的”；
  - 应有很好的“可读性”；
  - 在算法是正确的前提下，算法的可读性是摆在第一位的；
  - 必须具有“健壮性”：指算法应对非法输入的数据作出恰当反映或进行相应处理，一般情况下，应向调用它的函数返回一个表示错误或错误性质的值。
  - 算法的效率：应考虑所设计的算法具有“高效率与低存储量”。

1.4 算法和算法分析

- 评价算法效率的方法：
  - 事后统计：将算法实现，测算其时间和空间开销。
    - 优点：准确的实测值 。
    - 缺点： 编写程序实现算法将花费较多的时间和精力； 所得实验结果依赖于计算机的软、硬件等环境因素。
  - 事前分析：对算法所消耗资源的一种估算方法。
- 算法分析：对算法所需要的计算机资源—时间和空间进行估算。
  - 时间复杂性（Time Complexity）
  - 空间复杂性（Space Complexity）

1.4 算法和算法分析

- 算法时间复杂度（性）分析：

和算法执行时间相关的因素：

① 算法选用的策略

② 问题的规模

③ 编写程序的语言

④ 编译程序产生的机器代码的质量

⑤ 计算机执行指令的速度

与计算机软件和硬件相关，不考虑

1.4 算法和算法分析

- 算法分析-时间复杂度（性）：

算法的执行时间＝每条语句执行时间之和

每条语句执行次数之和

基本语句的执行次数

每条语句执行时间＝执行次数×执行一次的时间

单位时间

指令系统、编译的代码质量

- 执行一次的时间对于不同的算法来说都是一样的，所以一般忽略不计

1.4 算法和算法分析

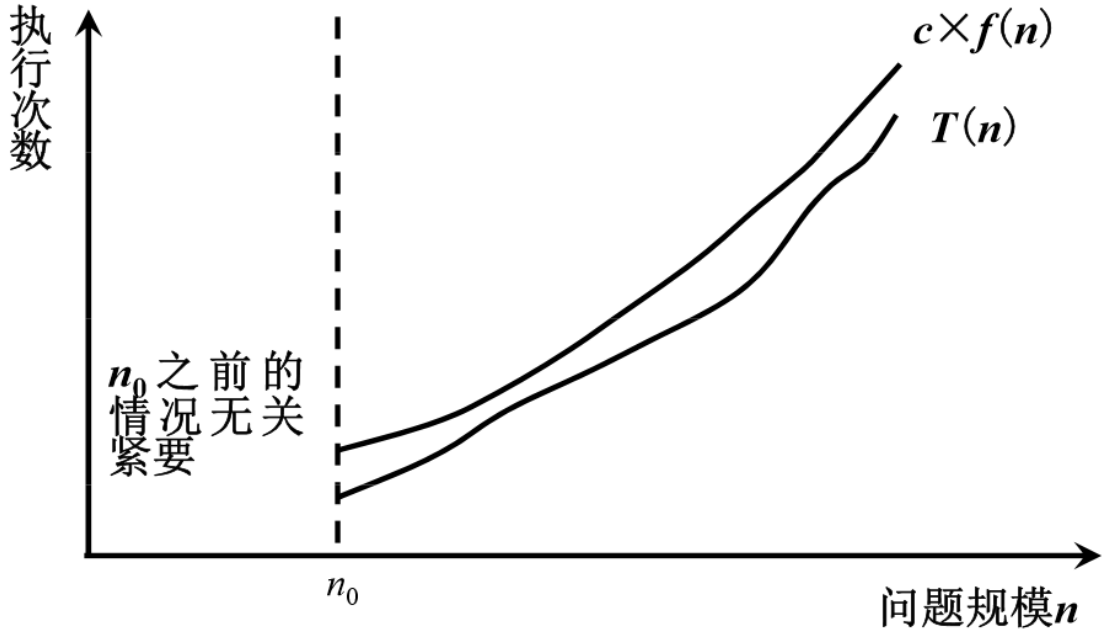
- 算法分析-时间复杂度（性）：
  - 算法的执行时间，是基本（操作）语句重复执行的次数，它是问题规模的一个函数。我们把这个函数的渐近阶称为该算法的时间复杂度。
  - 问题规模：输入量的多少。
  - 基本语句：是执行次数与整个算法的执行次数成正比的操作指令。

例：
```
for (i=1; i<=n; i++)
    for (j=1; j<=n; j++)
        x++;
```

- 问题规模：n
- 基本语句：x++
- 时间复杂性：$O(n^2)$

1.4 算法和算法分析

- 算法分析---大O符号：
  - 定义 若存在两个正的常数 c 和 n_0，对于任意 $n \geq n_0$，都有 $T(n) \leq c \times f(n)$，则称 $T(n) = O(f(n))$。



当问题规模充分大时在渐近意义下的阶

1.4 算法和算法分析

- 算法分析---大*O*符号：
  - 定理 若$A(n)=a_m n^m+a_{m-1}n^{m-1}+\ldots+a_1 n+a_0$是一个m次多项式，则 $A(n)=O(n^m)$。
  - 说明：在计算算法时间复杂度时，可以忽略所有低次幂（低阶）项 和高次幂（高阶）项的系数。

例：
```
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];
    }
```

问题规模：n

基本语句：*

复杂度：$O(n^3)$

1.4 算法和算法分析

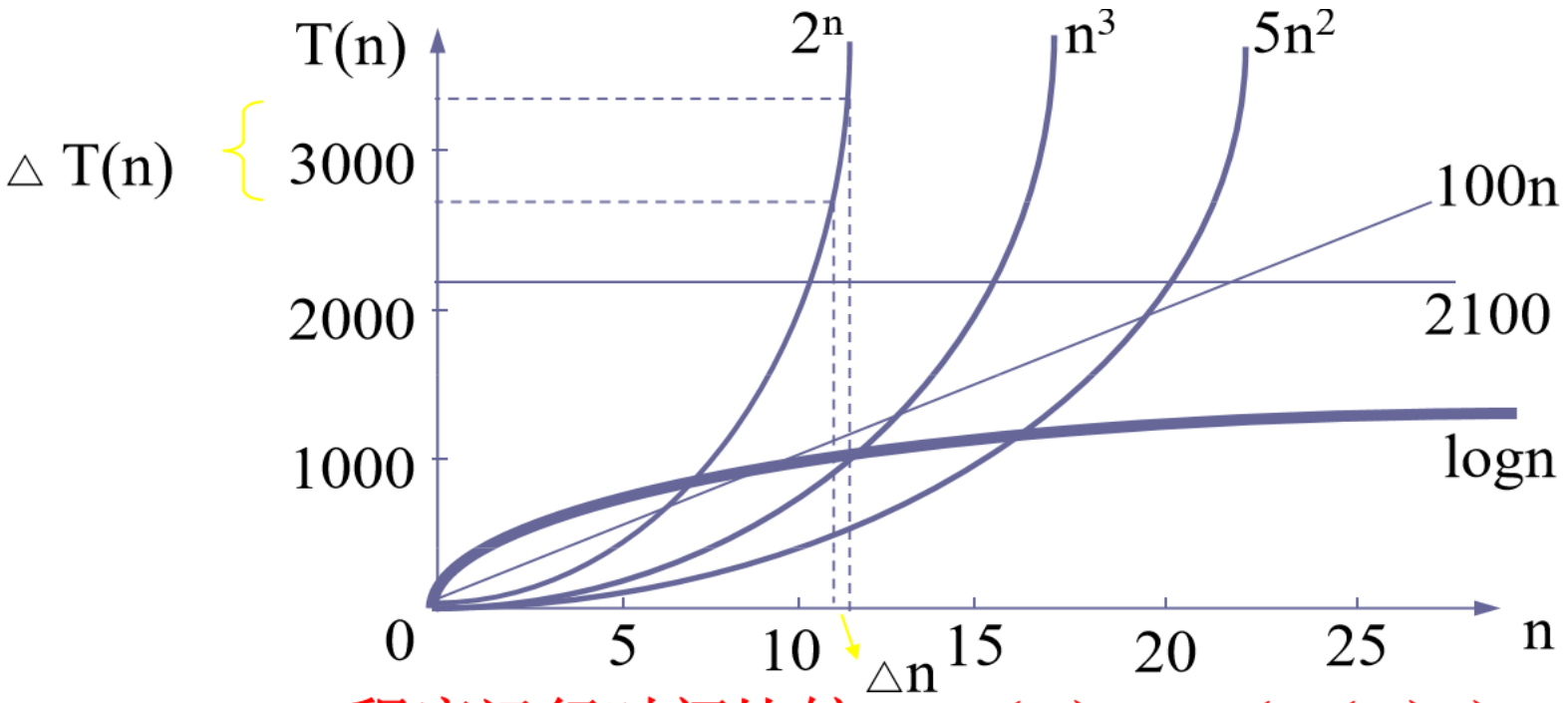
- 算法分析---好情况、坏情况、平均情况：
  - 例：在一维整型数组A[n]中顺序查找与给定值k相等的元素（假设该数组中有且仅有一个元素值为k）。

```
int Find(int A[ ], int n)
{
    for (i=0; i<n; i++)
        if (A[i]= =k) break;
    return i;
}
```

  - 基本语句的执行次数是否只和问题规模有关？
  - 结论：如果问题规模相同，时间代价与输入数据（的分布）有关，则需要分析最好情况、最坏情况、平均情况。

1.4 算法和算法分析

- 算法分析---常见的时间复杂度：



程序运行时间比较 T（n）=O（f（n））

– 常见阶的比较：

$$O(1)<O(\log_2 n)<O(n)<O(n\log_2 n)<O(n^2)<O(n^3)<\ldots<O(2^n)<O(n!)$$

1.4 算法和算法分析

- 算法分析---时间复杂度分析的基本方法：
  - 时间复杂性的运算法则

设$T_1(n)=O(f(n))$，$T_2(n)=O(g(n))$，则

- ①加法规则：$T_1(n)+T_2(n)=O(\max\{f(n), g(n)\})$
- ②乘法规则：$T_1(n)*T_2(n)=O(f(n)\cdot g(n))$

  - 时间复杂性的分析方法
    - 首先求出程序中各语句、各模块的运行时间，
    - 再求整个程序的运行时间。

1.4 算法和算法分析

- 算法分析---各种语句和模块分析应遵循的规则：

① 赋值语句或读/写语句:

- 运行时间通常取O(1)。有函数调用的除外，此时要考虑函数的执行时间。

② 语句序列:

- 运行时间由加法规则确定，即该序列中耗时最多的语句的运行时间。

③ 分支语句：

- 运行时间由条件测试（通常为O(1)）加上分支中运行时间最长的语句的运行时间

1.4 算法和算法分析

- 算法分析---各种语句和模块分析应遵循的规则：

④ 循环语句：

– 运行时间是对输入数据重复执行n次循环体所耗时间的总和 。

– 每次重复所耗时间包括两部分：一是循环体本身的运 行时间；二是计算循环参数、测试循环终止条件和跳 回循环头所耗时间。后一部分通常为O(1)。

– 通常，将常数因子忽略不计，可以认为上述时间是循 环重复次数n和m的乘积，其中m是n次执行循环体当中 时间消耗多的那一次的运行时间(乘法规则) 。

– 当遇到多重循环时，要由内层循环向外层逐层分析。 因此，当分析外层循环的运行时间时，内层循环的运 行时间应该是已知的。此时可以把内层循环看成是外层循环的循环体的一部分。

1.4 算法和算法分析

- 算法分析---各种语句和模块分析应遵循的规则：

⑤ 函数调用语句:

a) 若程序中只有非递归调用，则从没有函数调用的被调函数开始，计算所有这种函数的运行时间。然后考虑有函数调用的任意一个函数P，在P调用的全部函数的运行时间都计算完之后，即可开始计算P的运行时间。

b) 若程序中有递归调用，则令每个递归函数对应于一个未知的时间开销函数T(n)，其中n是该函数参数的大小，之后列出关于T的递归方程并求解之。

1.4 算法和算法分析

例 分析下述“选择”排序程序的时间复杂性：

```
void select_sort(int a[], int n)
 { //将a中整数序列重新排列成自小到大有序的整数序列
     for (i=0;i<n-1;++i) {
        j=i;
        for (k=i+1;k<n;++k)
            if (a[k]<a[j])  j=k;
            if (j != i) a[j]←→a[i] ;
         }
}//select_sort
```

基本操作：比较（数据元素）操作

时间复杂度： $O(n^2)$

1.4 算法和算法分析

例 编写求n!的程序，并分析其时间复杂性：

- 求n!的递归算法

```
long  fact ( int  n)
{   if ( n==0 ) || ( n ==1 )
        return( 1 );
    else
        return( n * fact(n – 1));
}
```

- 时间复杂性的递归方程

$$T(n)=\begin{cases} C & \text{当 } n=0,\ n=1 \\ G+T(n-1) & \text{当 } n>1 \end{cases}$$

- 解递归方程：

$$\begin{aligned} T(n) &= G+T(n-1) \\ T(n-1) &= G+T(n-2) \\ T(n-2) &= G+T(n-3) \\ &\cdots\cdots \\ T(2) &= G+T(1) \\ +\ T(1) &= C \\ \hline T(n) &= G(n-1)+C \end{aligned}$$

⇓

取 $f(n)=n$

$\therefore T(n)=O(f(n))$

$=O(n)$

1.4 算法和算法分析

- 算法分析---空间复杂性
  - 算法的空间复杂性是指算法在执行过程中的存储量需求
  - 一个算法的存储量需求除了存放算法本身所有的指令、 常数变量和输入数据外，还包括对数据进行操作的工 作单元和存储实现计算所需信息的辅助空间
  - 算法的存储量需求与输入的规模、表示方式、算法采用的数据结构、算法的设计以及输入数据的性质有关
  - 算法的执行的不同时刻，其空间需求可能是不同的
  - 算法的空间复杂性是指算法在执行过程中的最大存储量需求
  - 空间复杂性的渐近表示----空间复杂度

    T（n）= O（f（n））　其中，n为问题的输入规模

本章小结

- ✓ 数据结构的地位；
- ✓ 数据结构和算法等基本概念；
- ✓ 数据的逻辑结构： 线性结构和非线性结构；
- ✓ 数据的存储结构可以用以下存储方法；
  - 顺序存储
  - 链式存储
  - 索引存储
  - 散列存储
- ✓ 抽象数据类型
- ✓ 算法的特性
- ✓ 算法的复杂度分析