数据结构与算法分析

北京工业大学软件学院
沈琦
67396121-18
sheng@bjut.edu.cn

一、课程地位

- 《数据结构》是软件工程专业本科生的专业基础课程之一。
- 本课程不仅为操作系统、数据库系统、编译方法、计算机网络等后续课程提供了必要的知识基础，而且也为计算机及其专业人员提供了必要的技能训练。
- 增强求解复杂问题的能力

二、课程要求

- 知识方面：从数据结构的类定义和对象的使用，以及存储表示和操作的实现两个层次
  - 系统地学习和掌握常用的基本数据结构及其不同的实现
  - 掌握分析、比较和选择不同数据结构、不同存储结构、不同算法的原则和方法
  - 为后续课程的学习打好基础。
- 技能方面：系统地学习和掌握对象、类的设计方法和面向对象的程序设计风格，在不同的存储结构上实现算法的设计思想，从中体会算法设计的方式及技巧，提高分析问题和解决问题的能力。

- 实践性强，因此，除了理论学习，要重视实验教学环节。
- 课程形式：讲课、习题课、课堂讨论、自学、作业、上机实验。
- 学习要求：按时上课，认真听讲、阅读参考书、整理笔记、思考，讨论，提问。
- 成绩评定：

期末考试	70%
上机实验	20%
平时（考勤+作业）	10%

三、教材与参考书

- 数据结构（用面向对象方法与C++语言描述）

  殷人昆主编，清华大学出版社

- 《数据结构与算法》

  许卓群、杨冬青、唐世渭、张铭，高等 教育出版社

- 数据结构、算法与应用——C++语言描述

  （美）SARTAJ SAHNI，机械工业出版社

- 数据结构与算法（英文影印版）

  Jeffrey D. Ullman 等，清华大学出版社

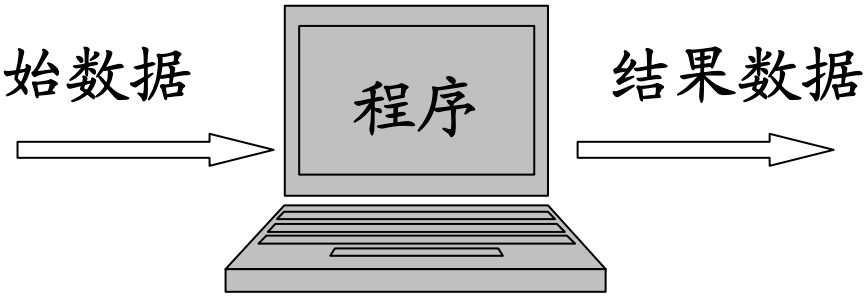
四、课程学习指导

- 在学习本课程之初，必须注意复习在用C / C++ 程序设计语言编写程序时的语法规则和方法。
- 在学习《数据结构》时，要注意知识体系。
- 注意复习、比较。
- 注意循序渐进。
- 算法思路的理解。
- 努力培养算法设计的能力。
- 最后强调的是，学习方法主要靠自己摸索，多总结，多思考，勤上机，勤交流，是把数据结构这门课程真正学好的关键。

1.1基本概念

计算机的发展

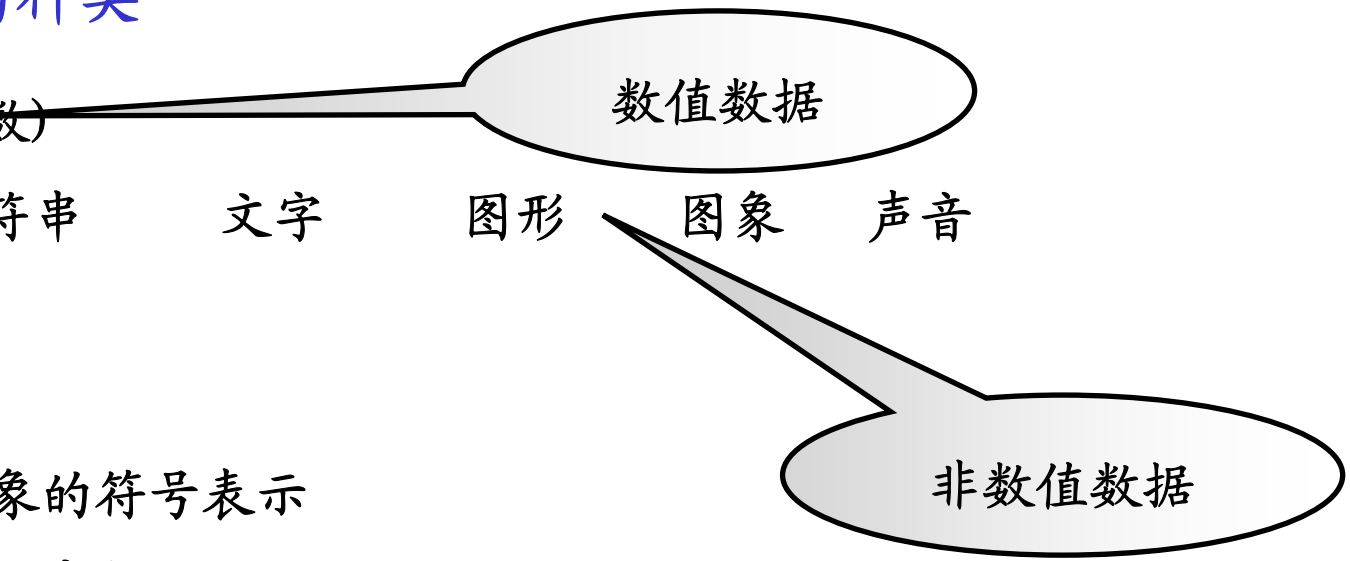
软件　　硬件　　应用领域



数据处理的种类

数（整数，实数）

字符　　字符串　　文字　　图形　　图象　　声音



数据

数据：客观对象的符号表示

数学中的整数、实数，
课程名，地名、书名

数值问题与非数值问题

1）数值问题

例1　已知：游泳池的长len和宽wide，求面积area

◆ 建模型：

问题涉及的对象：游泳池的长len 宽wide，面积area;

对象之间的关系：area=len×wide

◆设计 求解问题的方法

◆ 编程

```
main ( )
{     int len, wide ,area ;
      scanf (“%d %d%\n”, &l,&w);
      area=len*wide ;
      printf (“area=%d”,area);
}
```

例一“学生”表格

	学　号	姓　名	性 别	籍　贯	出 生 年 月
1	98131	刘 激 扬	男	北　京	1979.12
2	98164	衣 春 生	男	青　岛	1979.07
3	98165	卢 声 凯	男	天　津	1981.02
4	98182	袁 秋 慧	女	广　州	1980.10
5	98203	林 德 康	男	上　海	1980.05
6	98224	洪　伟	男	太　原	1981.01
7	98236	熊 南 燕	女	苏　州	1980.03
8	98297	宫　力	男	北　京	1981.01
9	98310	蔡 晓 莉	女	昆　明	1981.02
10	98318	陈　健	男	杭　州	1979.12

“课程”表格

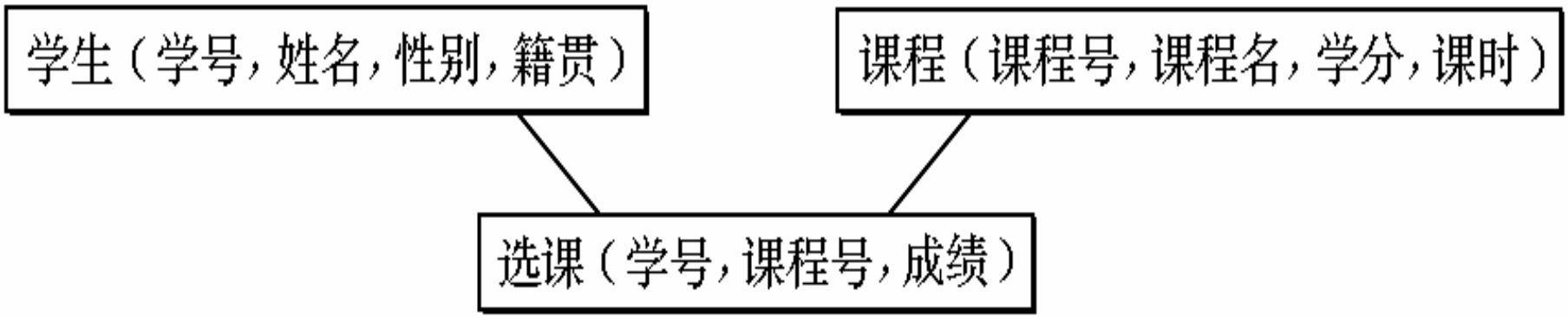
课程编号	课 程 名	学时
024002	程序设计基础	64
024010	汇编语言	48
024016	计算机原理	64
024020	数据结构	64
024021	微机技术	64
024024	操作系统	48
024026	数据库原理	48

例二 在学生选课系统中，一个学生可以选修多门课，一门课可以被多个学生选修，在“学生”和“课程”实体之间形成多对多的关系。

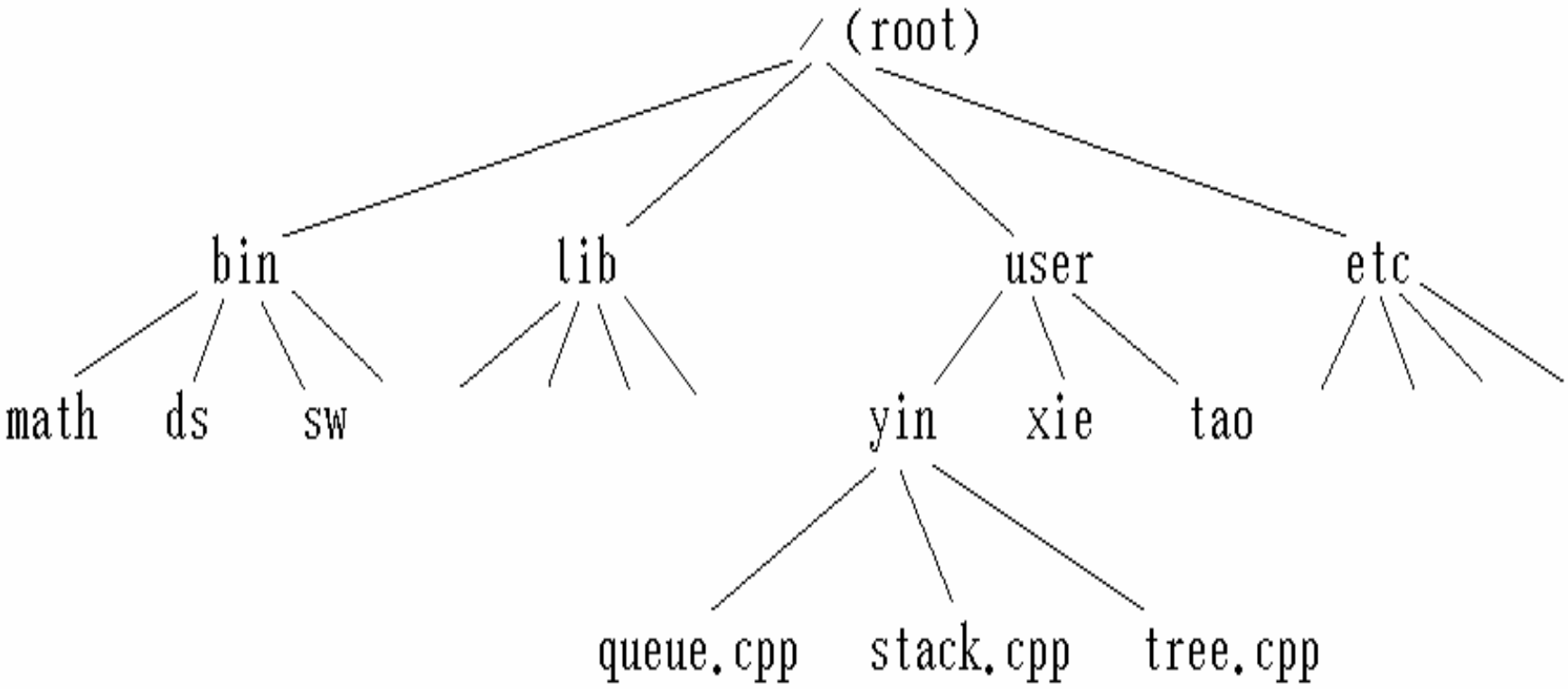
选课单包含如下信息：

学 号　　课程编号　　成 绩　　时 间

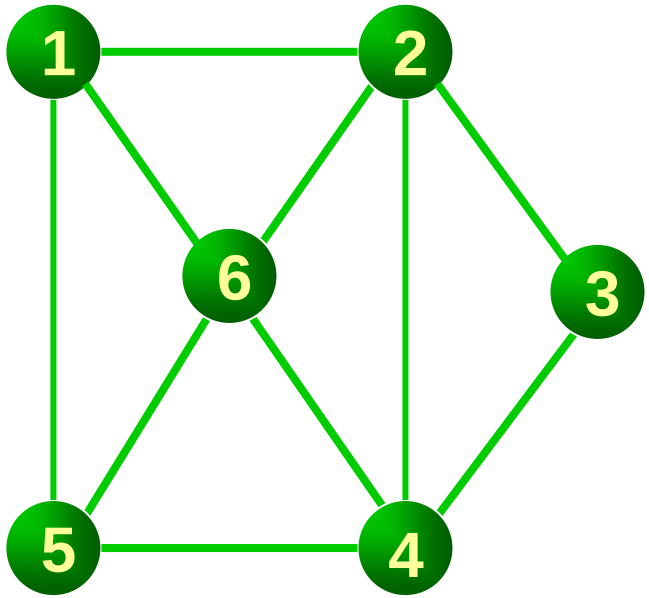
学生选课系统中实体构成的网状关系



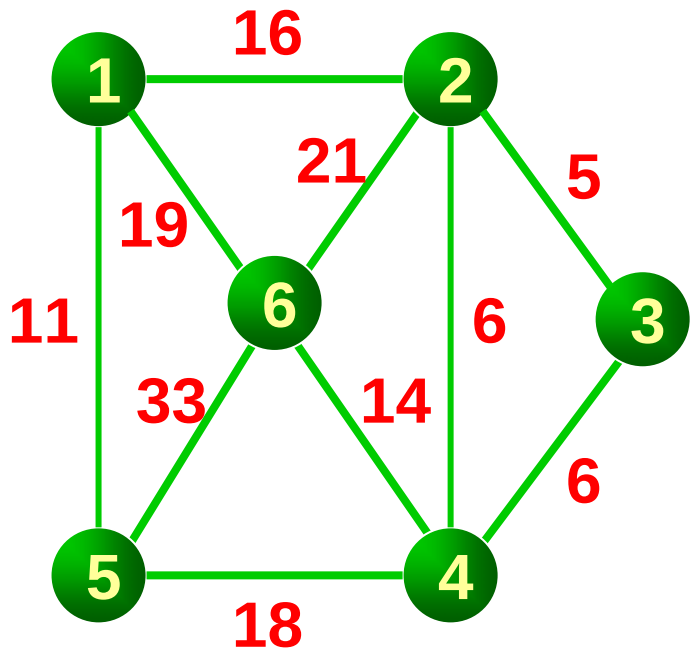
例三 一个典型的UNIX文件系统的层次结构



N 个网点之间的连通关系



图结构



网络结构

- 数据结构的研究问题:
  - 非数值数据之间的结构关系，及如何表示，如何存储，如何处理。
- 本课程讨论的问题:

  应用中常用的几种数据间的结构关系，及如何表示，如何存储，如何处理。

- **数据：**数据是信息的载体，是描述客观事物的数、字符、以及所有能输入到计算机中，被计算机程序识别和处理的符号的集合。
  - 数值性数据、非数值性数据
- **数据元素：**是数据（集合）中的一个“个体”，数据结构中讨论的基本单位。
- 数据项数据项：数据结构中讨论的最小单位，数据元素可以是数据项的集合

数据结构

- 定义：带结构的数据元素的集合。
- 记为：

  Data_Structure = {D, R, P}

- 其中：
  - D是数据对象
  - R是该对象中所有数据成员之间的关系的有限集合。
  - P 是对D的基本操作集。

- 数据结构： 带结构的数据元素的集合
- 数据结构是相互之间存在着某种逻辑关系的数据元素的集合。
- 例如
  - 一维数组{ a1, a2, a3, a4, a5, a6, a7,a8 }的数据元素之间存在如下的次序关系：

$$\{ \langle a_i, a_{i+1} \rangle \mid i=1, 2, 3, 4, 5, 6, 7\}$$

对每种数据结构，主要讨论如下三方面的问题：

1） 数据的逻辑结构；

2） 数据的存储结构；

3） 数据的运算，即对数据施加的操作

数据的逻辑结构：

逻辑结构是指从解决问题的需要出发，为实现必要的功能所建立的数据结构，它属于用户的视图，是面向问题的，是根据问题所要实现的功能建立的。数据元素之间的逻辑关系，是具体关系的抽象。

数据的存储结构（物理结构）：

物理结构是逻辑结构的物理存储方式，属于具体实现的视图，是面向计算机的，根据问题所要求的响应速度、处理时间、存储空间等建立的，是逻辑结构的存储映像。

数据的运算：

定义在数据的逻辑结构上的，抽象的操作。“做什么”

1.2 抽象数据类型

- 1.2.1 数据类型
- 定义：一组性质相同的值的集合，以及定义于这个值集合上的一组操作的总称.
- C语言中的数据类型
  - char int float double void

    字符型 整型 浮点型 双精度型 无值
- 数据类型是一种性质相同的值的集合以及定义于这个集合上的一组操作的总称。

1.2.2 数据抽象和抽象数据类型 (ADTs: Abstract Data Types)

- 抽象的本质就是抽取反映问题本质的东西，忽略非本质的细节。
- 抽象数据类型通常是指:
  - 由用户定义，用以表示应用问题的数据模型
  - 由基本的数据类型组成，并包括一组相关的服务（或称操作）
  - 信息隐蔽和数据封装，使用与实现相分离

自然数的抽象数据类型定义

ADT *NaturalNumber* is

objects: 一个整数的有序子集合,它开始于0,结束于机器能表示的最大整数(*MaxInt*)。

Function: 对于所有的 *x*, *y* ∈ *NaturalNumber*;*False*, *True* ∈ *Boolean*, +、-、<、==、=等都是可用的服务。

Zero( ) : *NaturalNumber* 返回自然数0

IsZero(*x*) : *Boolean* if (*x*==0) 返回 *True* else 返回 *False*

Add (*x*, *y*) : *NaturalNumber* if (*x*+*y*<=*MaxInt*) 返回 *x*+*y* else 返回 *MaxInt*

Subtract (*x*, *y*) : *NaturalNumber* if (*x* < *y*) 返回 0 else 返回 *x* – *y*

Equal (*x*, *y*) : *Boolean* if (*x*==*y*) 返回 *True* else 返回 *False*

Successor (*x*) : *NaturalNumber* if (*x*==*MaxInt*) 返回 *x* else 返回 *x*+1

end *NaturalNumber*

1.3 数据结构的抽象层次

- 线性结构
  - 直接存取类　数组，文件
  - 顺序存取类　表，栈，队列，优先队列
  - 广义索引类　线性索引，搜索树
- 非线性结构
  - 层次结构类　树，二叉树，堆
  - 群结构类　集合，图

1.3.1 线性结构

线性结构又称线性表，类中所有数据成员都按照某种次序排列在一个序列中，如图所示。

更一般地可以表示成：$a_1, a_2, \ldots a_{i-1}, a_i, a_{i+1}, \ldots, a_{n-1}, a_n$

对于线性聚类中的每一个元素：

a_1 ： 有且仅有一个后继，没有前驱；

a_i ： 有且仅有一个前去和一个后继；

a_n ： 有且仅有一个前驱，没有后继。

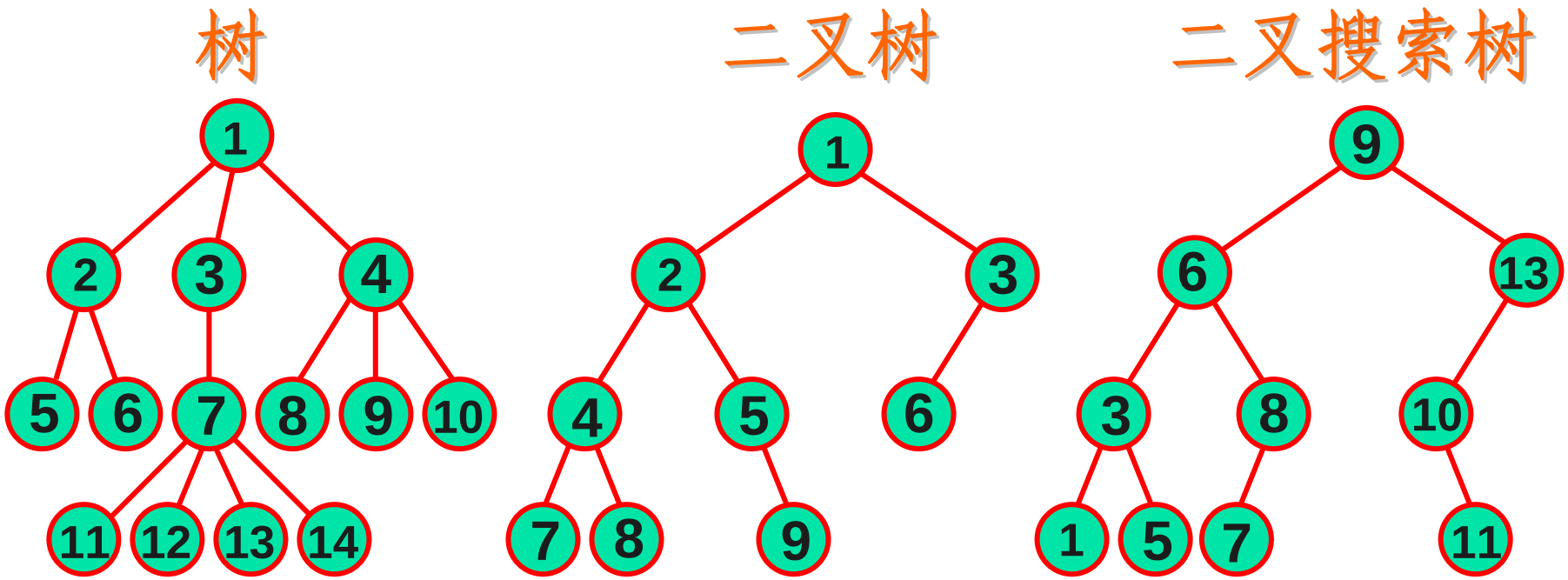
1.3.2 非线性聚类

前驱：除根结点之外，

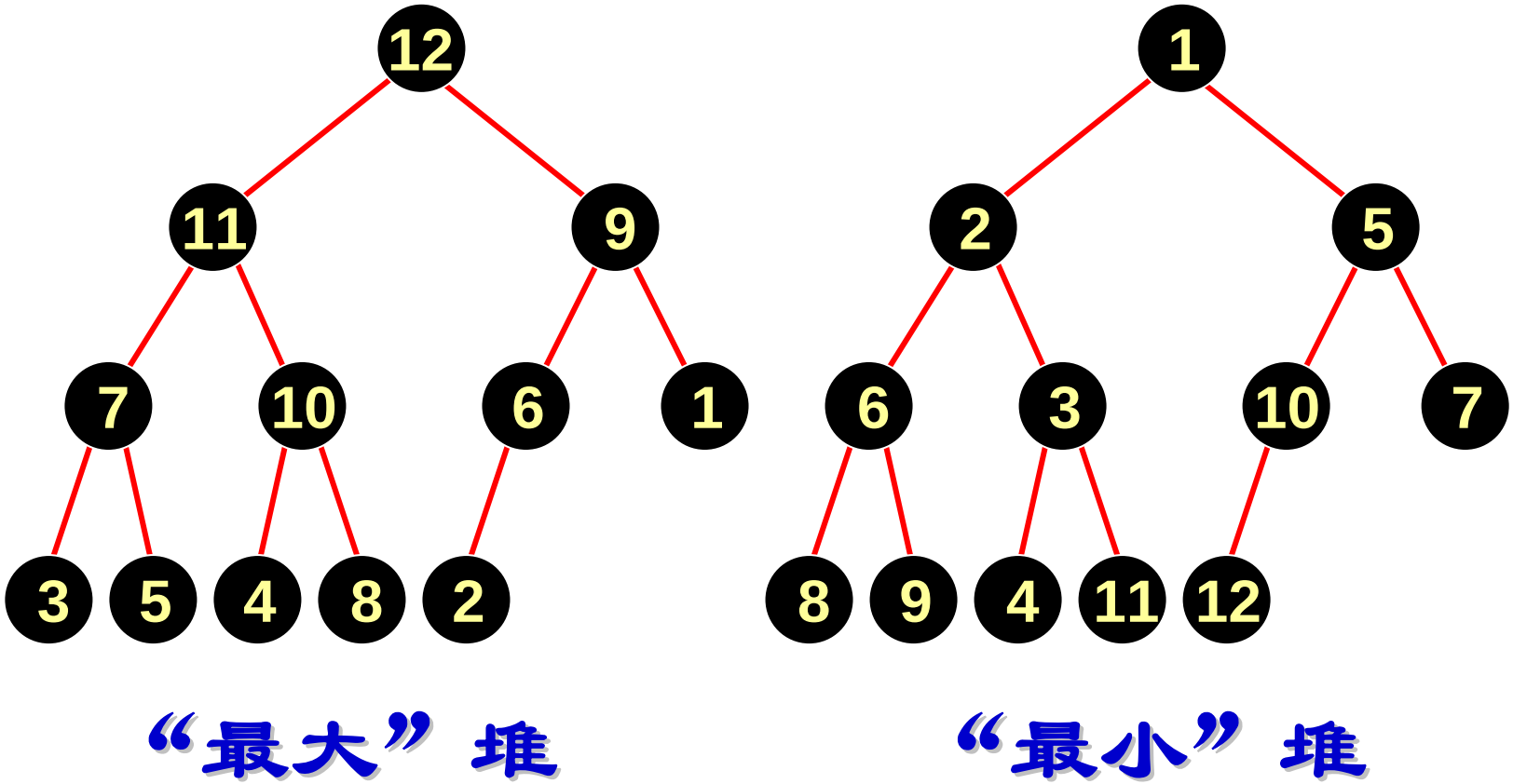
后继：除叶子结点之外，

- 树形结构

树形结构最重要的特性是层次性。



堆结构



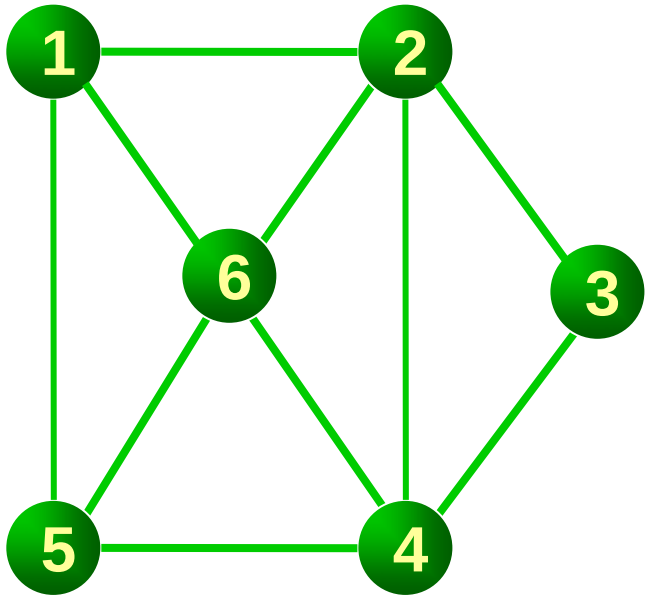
“最大”堆　　“最小”堆

➢ 群聚集类

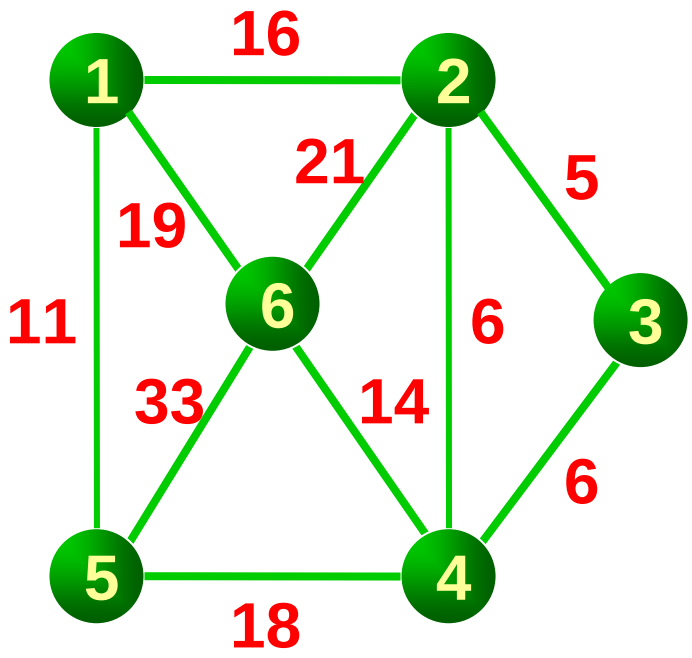
群聚集类的特点是群聚集中所有元素之间没有任何顺序关系。

集合中的数据元素之间是无序，且没有重复的元素。

图也是一种群聚集类。



图结构



网络结构

1.5 算法定义

算法是一个有穷的指令集，这些指令为解决某一特定任务规定了一个运算序列。

特性：

1 输入　有0个或多个输入

2 输出　有一个或多个输出(处理结果)

3 确定性　每步定义都是确切、无歧义的

4 有穷性　算法应在执行有穷步后结束

5 有效性　每一条运算应足够基本

算法设计　自顶向下，逐步求精

例一　将数组中的数据进行排序。

1 明确问题：非递减排序

2 解决方案：逐个选择最小数据

3 算法框架：

```
for ( int i=0; i<n-1; i++ )
{     //n-1趟
        从a[i]检查到a[n-1];
      若最小的整数在a[k]，交换a[i]与a[k];
}
```

4 细化　细化需要解决两个问题：选择最小数、两个数据的交换。

1.7 性能分析与度量

算法的性能标准

算法的后期测试

算法的事前估计

1.7.1算法的性能标准

1 正确性　要求算法能够正确执行预先规定的功能和性能要求。

2 可使用性 要求算法能够方便地使用（用户友好性）。

3 可读性 算法应当是可读的，这是理解、测试和修改算法的需要。

4 效率　算法的效率主要指算法执行时计算机资源的消耗，包括存储和运行时间的开销。

5 健壮性　具有容错性或例外处理，能够对不合理的数据进行检查。

1.7.2 时间复杂度度量

一个算法所用的时间等于编译时间加上运行时间。

编译时间与编译程序有关，与实例特性无关。

运行时间取决于下列因素：

1) 算法本身

2) 问题的规模

3) 所选用的语言

4) 编译程序所产生的机器代码的质量

5) 机器执行指令的速度

显然，同一个算法用不同的语言实现，或者用不同的编译程序进行编译，或者在不同的计算机上运行，效率均不相同。这表明使用绝对的时间单位衡量算法的效率是不合适的。因此，可以认为算法的运行时间只依赖于问题的规模，是问题规模的函数。

时间复杂度

例　矩阵的乘法

```
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];
    }
```

整个算法的执行时间与该基本操作（乘法）重复执行的次数n^3成正比，记作$T(n)=O(n^3)$.

一般地，算法中基本操作重复执行的次数是问题规模n的某个函数$f(n)$，算法的时间度量记作：

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

算法执行时间的增长率与$f(n)$的增长率相同，称作**算法的渐进时间复杂度**，简称**时间复杂度**。

语句的频度

语句的频度指的是该语句重复执行的次数。

例如：

```
1)    {++x; s=0}
2)    for  (a=1; a<=n; ++a) {++x; s+=x; }
3)    for  (b=1; b<=n; ++b)
            for  (c=1; c<=n; ++c)  {++x; s+=x; }
```

含基本操作“x加1”的语句频度分别为：1、n和n^2，则对应的时间复杂度分别为：O(1)、O(n)和$O(n^2)$，分别称为常数阶、线性阶和平方阶。此外，还有队数阶和指数阶等。

$$c < \log_2 n < n < n\log_2 n < n^2 < n^3 < 2^n < 3^n < n!$$

加法规则

针对并列程序段：

$$T(n, m) = T1(n) + T2(m)$$
$$= O(\max(f(n), g(m)))$$

乘法规则

- 针对嵌套程序段

$$T(n, m) = T1(n) * T2(m)$$
$$= O(f(n) * g(m))$$

2 算法空间复杂度

设执行算法所需的辅助空间是问题规模n的某个函数g(n)，则算法空间复杂度记作：　S（n）= O(g(n))

例　计算 $f(x)=a_0+a_1x+a_2x^2+\ldots+a_nx^n$

解法1 先计算x 的幂，存于power[ ]中，再分别乘以相应的系数

```
# define N 100
 float   evaluate (float coef[ ], float x , int n )
{ float  power[N], f; int i;
  for (power[ 0]=1, i = 1; i<=n; i++ ) power[i]=x*power[i-
1];
        for (f = 0, i=0; i<=N; i ++) f=f+coef[i]*power[i];
 return(f);
}
```

问题规模为n，　算法时间复杂度: **O(n)**
空间复杂度：　**O(N)**

解法2　　$f(x)=a_0+(a_1+a_2+\ldots+(a_{n-1}+a_nx)x)..x)x$

```
# define N 100
 float   evaluate (float coef[ ], float x , int n )
{ float  f; int i;
    for (f = coef[n], i=n-1; i>=0; i --) f=f*x+coef[i];
 return(f);
}
```

时间复杂度为O(n). 空间复杂度为O(1)

重点及要点

- 基本概念：理解什么是数据、数据对象、数据元素、数据结构、数据的逻辑结构与物理结构、数据结构的抽象层次。
- 熟练掌握逻辑结构的两大类型：线性结构、非线性结构；物理结构的两大类型：顺序映像、非顺序映像。
- 要点：
  - 抽象数据类型的封装性
  - 面向对象=对象+类+继承+通信。
  - 面向对象系统结构的稳定性
  - 面向对象方法着眼点在于应用问题所涉及的对象。

4、算法与算法分析：理解算法的定义、算法的特性、算法的时间代价、算法的空间代价，熟练掌握时间复杂度的计算方法。

要点：

- 算法与程序的不同之处需要从算法的特性来解释
- 算法的正确性是最主要的要求
- 算法的可读性是必须考虑的
- 算法的事前估计
- 程序的时间代价是指算法的渐进时间复杂性度量

作业

- 1.4
- 1.6
- 1.7
- 1.10