数 据 结 构
（Data Structure)

2021.9

教学内容

第一章 绪论
第二章 线性表
第三章 栈和队列
第四章 串
第五章 数组和广义表
第六章 树和二叉树
第七章 图
第九章 查找
第十章 内部排序

注：加 * *的章节一般不讲

教学要求

- 学会分析研究计算机加工的数据结构的特性，为应用涉及的数据选择适当的逻辑结构、存储结构及相应算法，并初步掌握算法的时间分析和空间分析的技术；

- 进行一些复杂程序设计的训练，要求编写的程序结构清楚和正确易读，符合软件工程规范。

第1章 绪论

- **1.1** 数据结构
- **1.2** 基本概念和术语
- **1.3** 抽象数据类型
- **1.4** 算法和算法分析

引 论

- 对于一个课题，在计算机领域，一般遵循下面的解决原则：

  需求分析→总体设计→ 模块分割→建立数学模型

  → 解数学模型的算法→ 程序编制→调试→ 结果

- 数据结构涉及到：数学模型的建立和对该模型具体实现的对应的算法。

例1、电话号码查询系统

设有一个电话号码薄，它记录了n个人的名字和其相应的电话号码，假定按如下形式安排：

$(a_1，b_1)(a_2，b_2)…(a_n，b_n)$

其中a_i，b_i(i=1，2…n) 分别表示某人的名字和对应的电话号码。要求设计一个算法，当给定任何一个人的名字时，该算法能够打印出此人的电话号码，如果该电话簿中根本就没有这个人，则该算法也能够报告没有这个人的标志。

姓名	电话号码
陈海	13612345588
李四锋	13056112345
。。。	。。。

算法的设计，依赖于计算机如何存储人的名字和对应的电话号码，或者说依赖于名字和其电话号码的结构。

数据的结构，直接影响算法的选择和效率。

上述的问题是一种数据结构问题。可将名字和对应的电话号码设计成：二维数组、表结构、向量。

假定名字和其电话号码逻辑上已安排成n元向量的形式，它的每个元素是一个数对$(a_i，b_i)$，$1≤i≤n$

数据结构还要提供每种结构类型所定义的各种运算的算法。

例2、图书馆的书目检索系统自动化问题

建立一张按登录号（关键字）顺序排列的书目文件和一些按书名、作者名和分类号顺序排列的索引表。

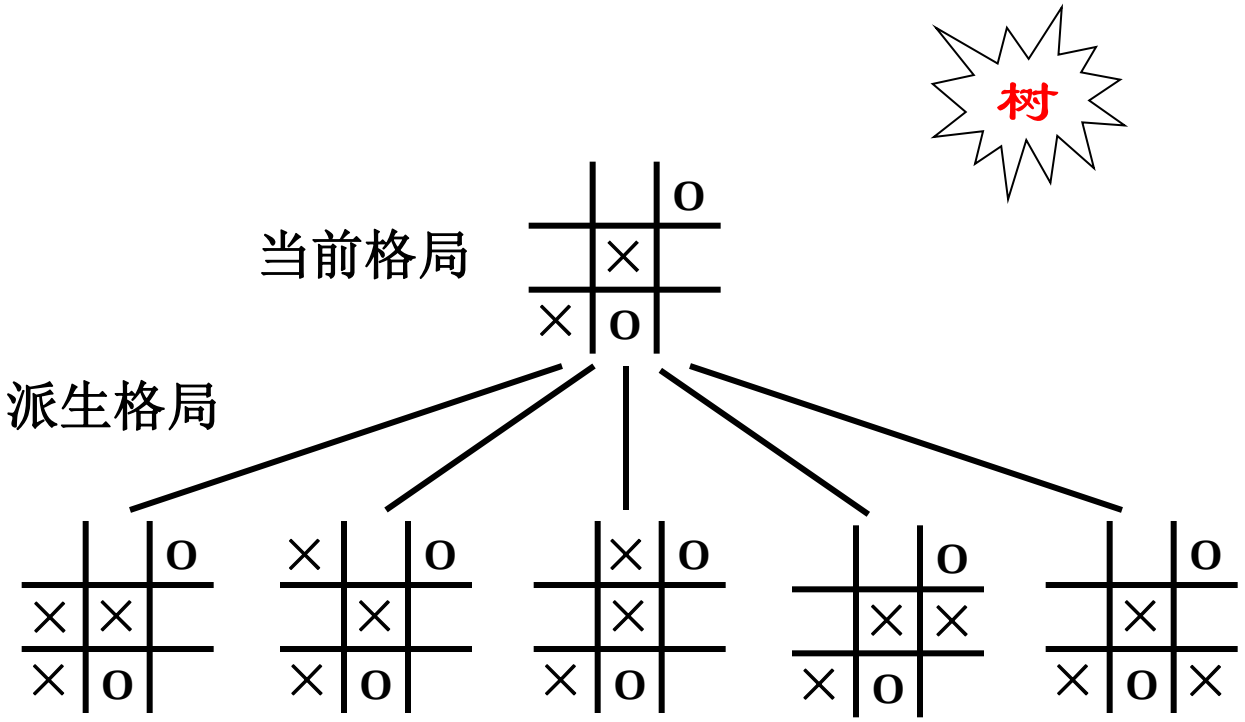
由这4张表构成的文件便是书目自动检索的数学模型。计算机的主要操作按照某个特定要求对书目文件进行查询

001	数据结构	周劲	S01	…
002	程序设计	潘玉奇	L01	…
003	数据结构	王永燕	S01	…
004	数据库	曲守宁	D01	…
⋮	⋮	⋮	⋮	⋮

周劲	001
潘玉奇	002
王永燕	003
曲守宁	004

数据结构	001,003
程序设计	002
数据库	004
⋮	

例3、计算机和人对奕问题



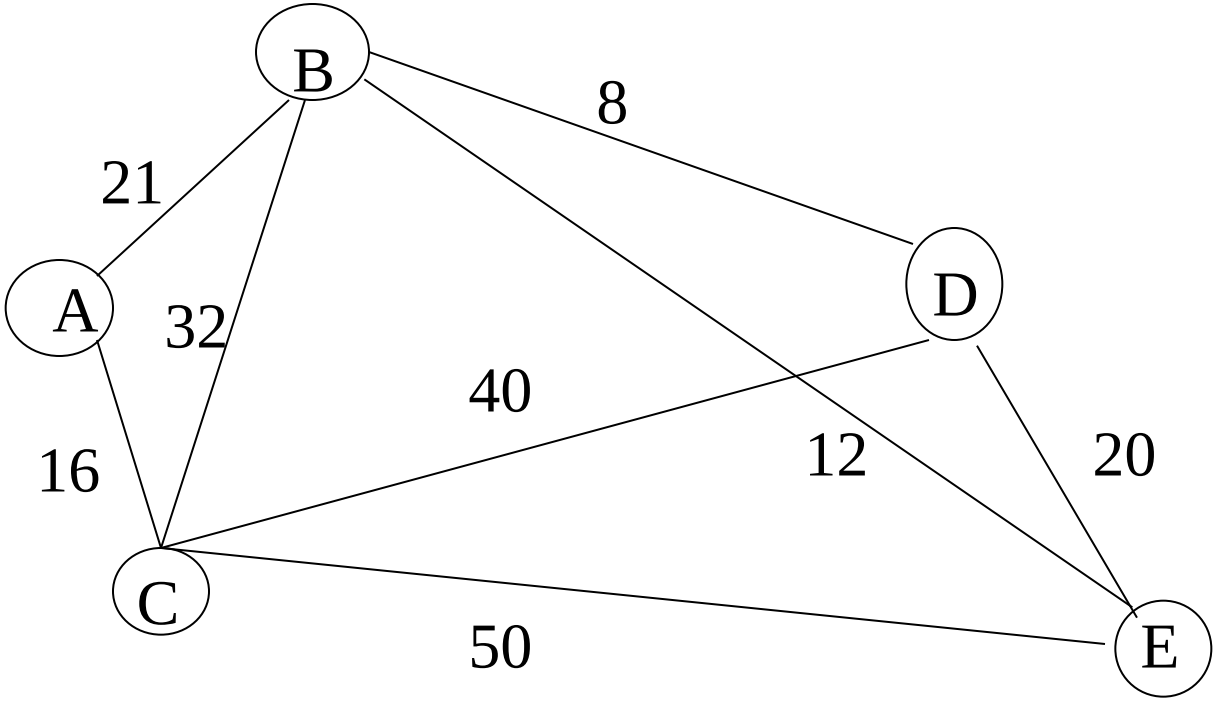
例4、居民区铺设煤气管道问题

这个问题的数学模型可以用“图”来表示，图中顶点表示居民区，顶点之间的连线及它上面的数值表示可以架设的管道及所需经费。

假设有n个居民区，求解的算法为：在可能架设的m条管道中选取n-1条，既能连通n个居民区，又使总投资达到最小。

这个问题实际上是“求图的最小生成树”的问题。

管道铺设问题



通过以上几例可以直接地认为：

数据结构就是研究数据的逻辑结构和物理结构以及它们之间相互关系，并对这种结构定义相应的运算，而且确保经过这些运算后所得到的新结构仍然是原来的结构类型。

- 先行课：离散数学、C语言、程序设计基础等
- 数据结构的地位：数学、硬件、软件之间。核心专业基础课
- 数据结构是一般程序设计的基础，也是设计编译程序、操作系统、数据库系统及其他系统程序和大型应用程序的基础。

1.1数据结构的基本概念和术语

1. 基本术语

（1）数据：描述客观事物的数字、字符以及所有能输入到计算机中并被计算机程序处理的符号的集合。（数字、字符、声音、图形、图像等等）

（2）数据元素：数据的基本单位，在计算机程序中常常作为一个整体进行考虑和处理，如记录/结构。

（3）数据项：数据的不可分割的最小单位，如结构中的域。

（4）数据对象：性质相同的数据元素的集合，是数据的一个子集。例，整数数据对象

2. 数据结构

（1）定义：是相互之间存在一种或多种特定关系的数据元素的集合。

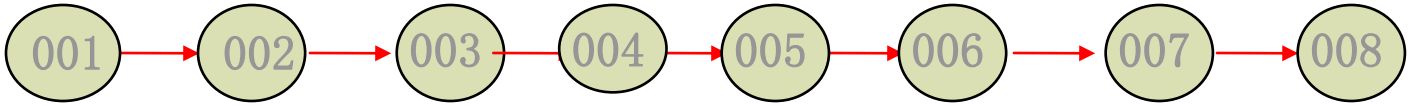
- 数据之间不是相互独立的，他们之间有某种特定的关系，这种数据元素之间的关系，称为“结构”

结构=关系+实体

- 另一种定义：按照逻辑关系组织起来的一批数据，按一定的存储方法把它存储在计算机中，并在这些数据上定义了一个运算的集合。

- 形式定义：二元组Data_Structure= (D,S) 其中D是数据元素的有限集，S是D上关系的有限集

例：学生基本情况表的二元组表示（D，S）

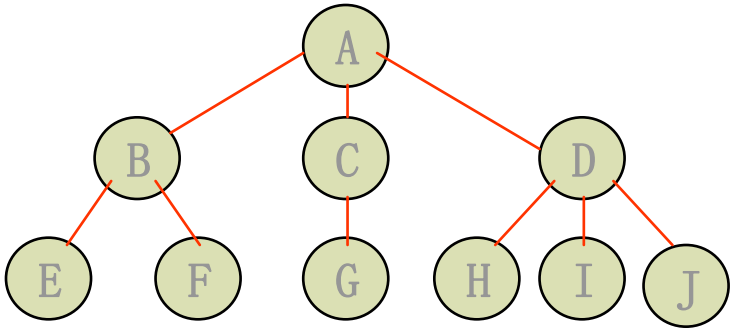


D = { 001，002，003，004，005，006，007，008}

S = { R }

R= {<001,002>,<002,003>,<003,004>,<004,005>,<005,006>,
<006,007>,<007,008> }

例：家族树的二元组表示（D，S）



D = { A，B，C，D，E，F，G，H，I，J}

S = { R }

R = { <A,B>,<A,C>,<A,D>,<B,E>,<B,F>,<C,G>,<D,H>,<D,I>,<D,J> }

❖例：　复数的数据结构定义如下：

Complex=(C，R)

其中：C是含两个实数的集合｛C1，C2｝，分别表示复数的实部和虚部。R={P}，P是定义在集合上的一种关系{〈C1，C2〉}。

❖例：编制一个事务管理程序，管理学校科研小组的各项事务。为科研小组设计一个数据结构。假设每个小组由一位教师、1~3名研究生及1~6名本科生组成，小组成员之间的关系是：教师指导研究生，每位研究生指导1~2名本科生。

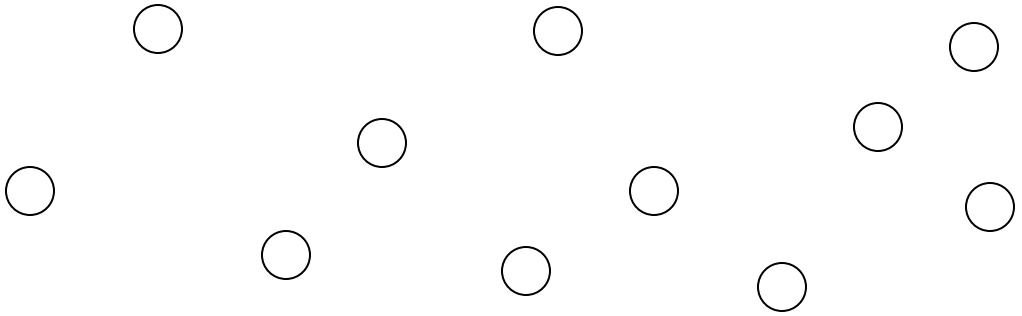
数据结构：Group=(P，R)

其中：$P=\{T，G_1，\ldots，G_n，S_{11}，\ldots，S_{nm}\}_{1\le n\le 3，\ 1\le m\le 2}$，$R=\{R_1，\ R_2\}$，$R_1=\{<T，G_i>\mid_{1\le i\le n，\ 1\le n\le 3}\}$，$R_2=\{<G_i，S_{ij}>\mid_{1\le i\le n，\ 1\le j\le m，1\le n\le 3，\ 1\le m\le 2}\}$

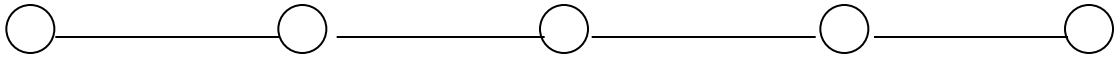
（2）四种基本结构（逻辑结构） p5

- 集合：元素仅属于同一个集体，没有其他关系。
- 线性结构：存在一对一 关系，序列相邻，次序关系。
- 树型结构：存在一对多关系，层次关系。
- 图状结构（网状结构）：存在多对多关系，任意性

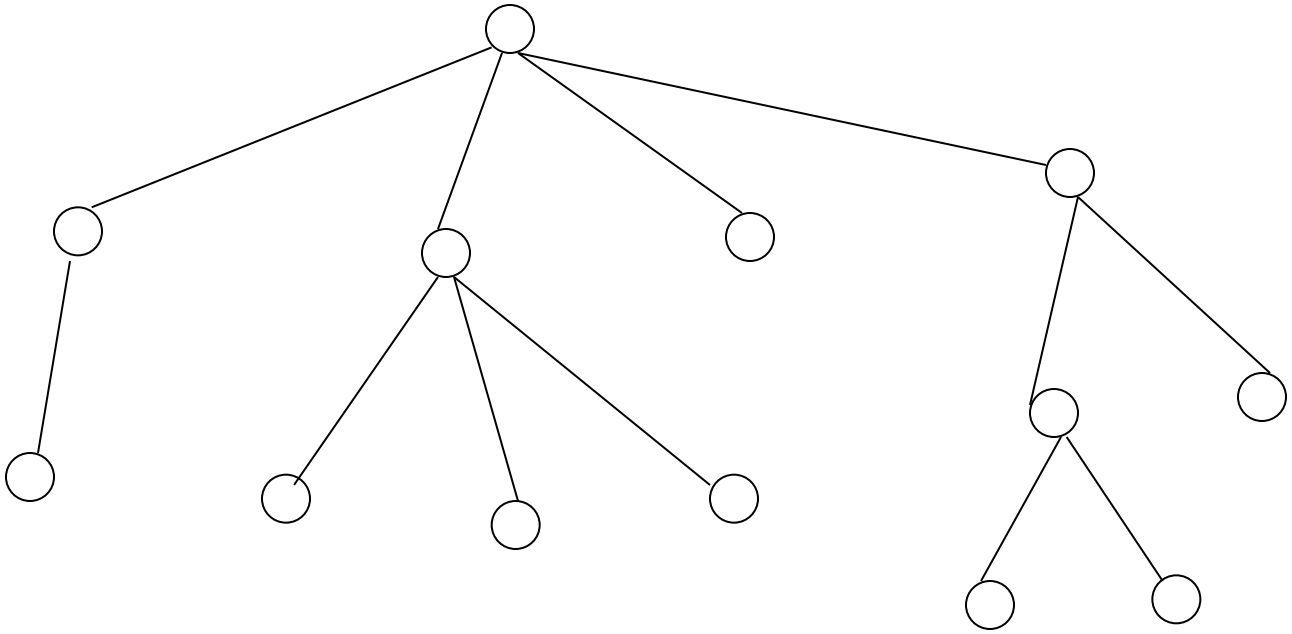
集合



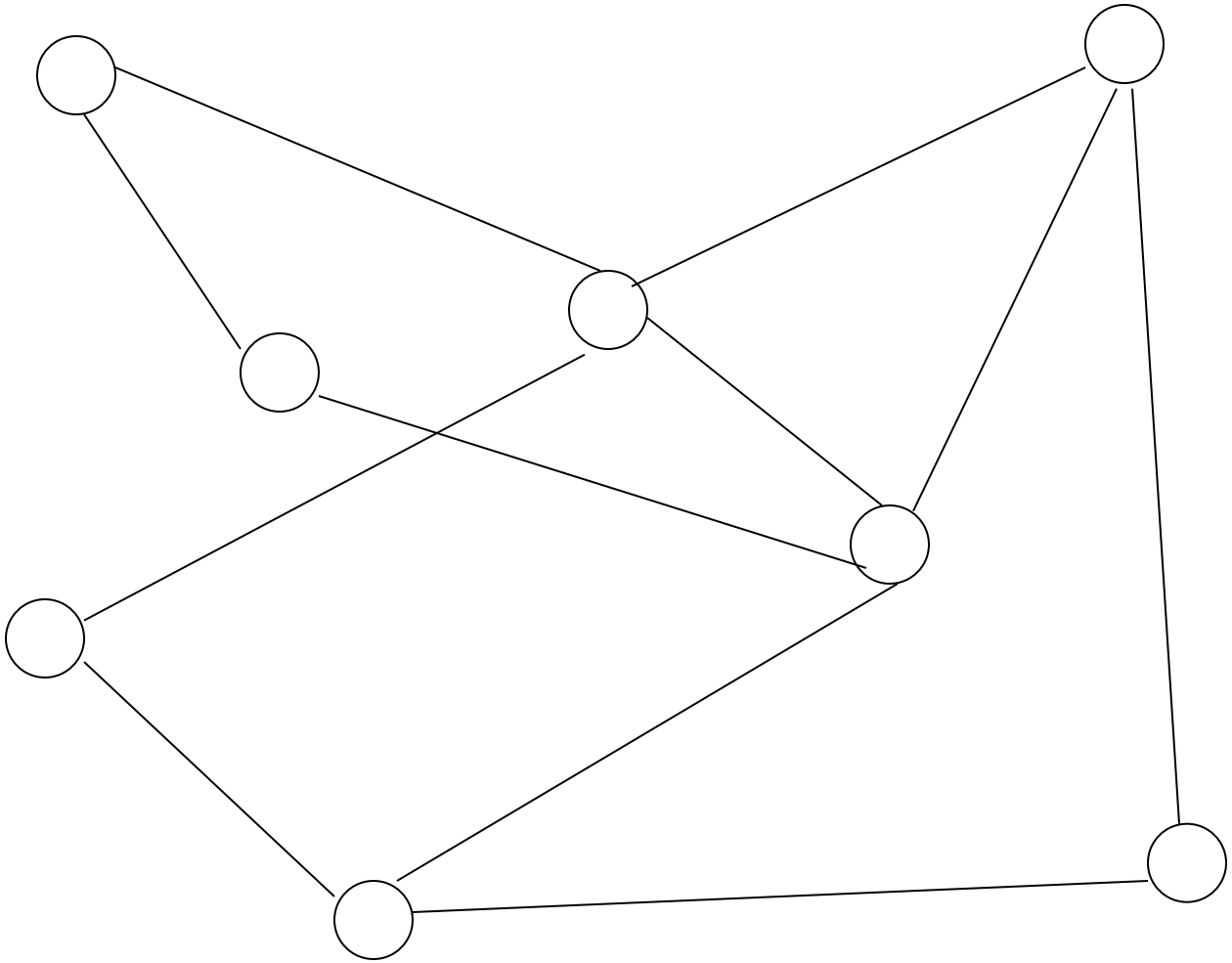
线性



树



图



实例：

表：计算机系人事表

工号	姓名	性别	职务	教研室	工作时间	发表论文
01			系主任	软件	1981.1	A,B
02			教研室主任	软件	1985.1	B,C,E,F
03			教师	软件	1990.8	C,D
04			教师	应用	1987.8	A,G
05			教师	应用	1975.9	E,I
06			教师	应用	1992.2	F,J
07			教师	软件	1983.8	D,L
08			教研室主任	应用	1986.7	G,H
09			教师	应用	1995.8	H,I,J,K
10			教师	软件	1989.2	L,K

- 线性结构示例 R={<05,01>, <01,07>, <07,02>, <02,08>, <08,04>, <04,10>, <10,03>, <03,06>,<06,09>}
- 树型结构示例
  R={<01,02>,<01,08>,<02,03>,<02,10>,<02,07>, <08,04>,<08,05>,<08,06>,<08,09>}
- 图状结构示例
  R={(01,02),(01,04),(02,05),(02,06),(02,03), (03,07),(04,08),(05,09),(06,09),(07,10),(08 ,09),(09,10)}

3. 数据结构的划分

（1）按数据结构的性质划分

- 数据的逻辑结构——数据元素之间的逻辑关系

（设计算法—— 数学模型）

- 数据的物理结构——数据结构在计算机中的映像

（存储结构，算法的实现）

- 物理结构就是逻辑结构到存储器的一个映射。
- 存储器模型：一个存储器M是一系列固定大小的存储单元，每个单元U有一个唯一的地址A(U)，该地址被连续地编码。每个单元U有一个唯一的后继单元U’=succ(U)
- 四种存储结构：顺序存储、链接存储、索引存储、散列存储

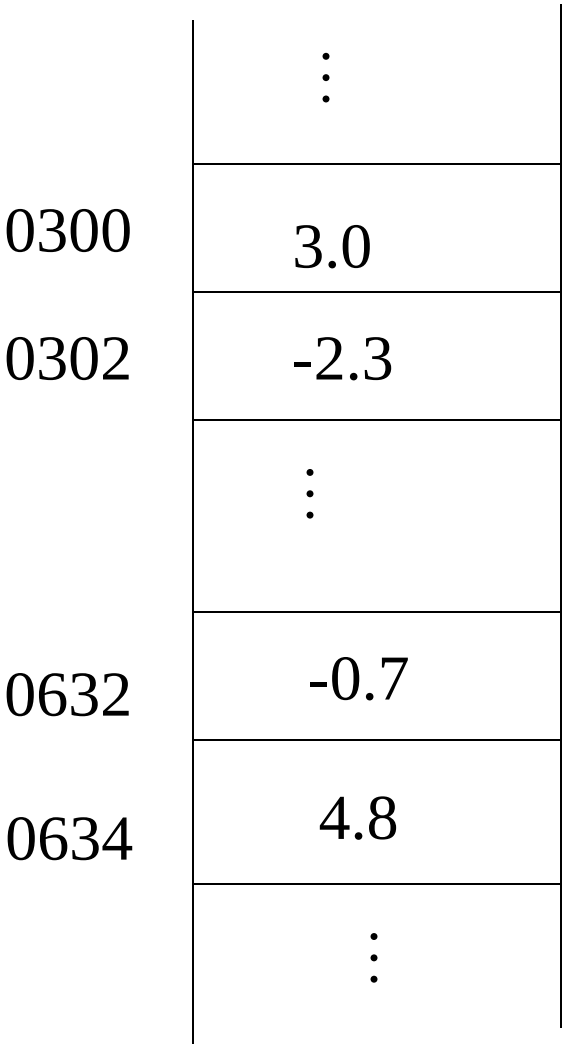
3. 数据结构的划分

（2）按数据结构在计算机内的存储方式来划分

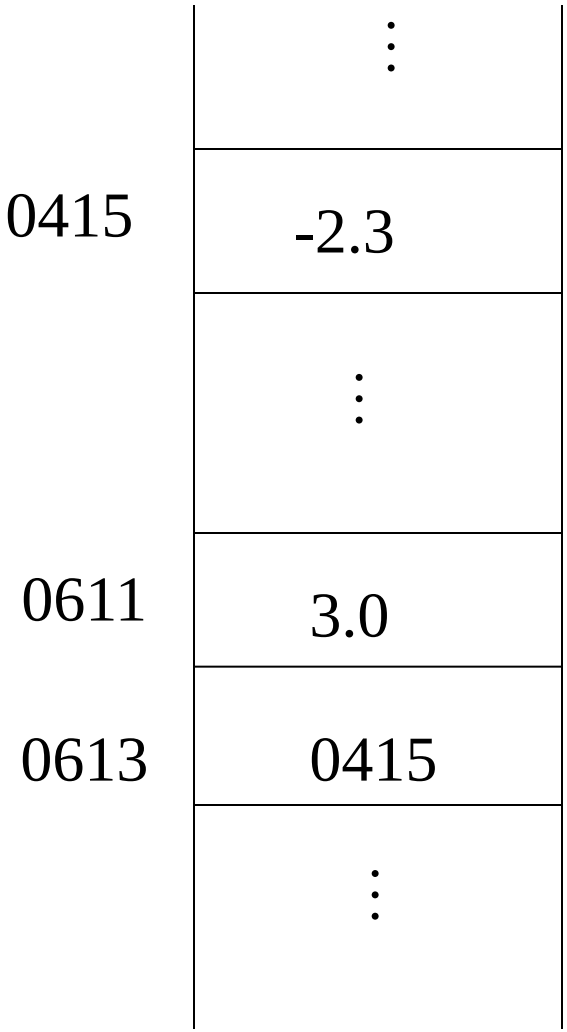
- 顺序存储结构——借助元素在存储器的相对位置来表示数据元素之间的逻辑关系。
- 链式存储结构——借助指示元素存储地址的指针表示数据元素之间的逻辑关系。

在C语言中，用一维数组表示顺序存储结构；用结构体类型表示链式存储结构。

例：表示复数z1=3.0-2.3*i*和z2=-0.7+4.8*i*的存储结构示意图



顺序存储结构



链式存储结构

3. 数据结构的划分

（2）按数据结构在计算机内的存储方式来划分

- 索引存储方法：在存储结点的同时，还建立附加 的索引表，索引表中的每一项称为索引项，形式为：关键字，地址。

- 散列存储方法：根据结点的关键字直接计算出该结点的存储地址。

说明：四种存储方法可结合起来对数据结构进行存储映像。

3. 数据结构的划分

（3）按数据结构的操作来划分

- 静态结构——经过操作后，数据的结构特征保持不变（如数组）。
- 半静态结构——经过操作后，数据的结构特性只允许很小变迁（如栈、队列）。
- 动态结构——经过操作后，数据的结构特性变化比较灵活，可随机地重新组织结构（如指针）。

逻辑结构与物理结构的关系

- 逻辑结构：描述数据元素之间的逻辑关系
- 讨论数据结构的目的是为了在计算机中实现对它的操作
- 存储结构：计算机中的表示
- 数据的逻辑结构与物理结构是密切相关的
- 算法的设计取决于选定的数据（逻辑）结构
- 算法的实现依赖于采用的存储结构

4. 数据的运算

数据的运算：就是施加于数据的操作，如查找、添加、修改、删除等。在数据结构中运算不仅仅实加减乘除这些算术运算，它的范围更为广泛，常常涉及算法问题。

举例：线性表的初始化、查找、插入、删除操作等

- 抽象运算定义在逻辑结构上，而实现在存储结构上。

- 综上所述，我们可以将数据结构定义为：
  按某种逻辑关系组织起来的一批数据，
  按一定的方式存储到计算机的存储器中，
  并在这些数据之上定义一个运算的集合，
  就称之为一个数据结构。

5. 数据类型

在用高级程序语言编写的程序中,必须对程序中出现的每个变量、常量或表达式,明确说明它们所属的数据类型。

不同类型的变量,其所能取的值的范围不同,所能进行的操作不同。**数据类型**是一个值的集合和定义在此集合上的一组操作的总称。

如：C/C++中的int就是整型数据类型。它是所有整数的集合(在16位计算机中为－32768到32767的全体整数)和相关的整数运算(如＋、－、＊、／等)。

5. 数据类型

数据类型是和数据结构密切相关的一个概念。 在C语言中数据类型有：基本类型和构造类型。

数据结构不同于数据类型，也不同于数据对象，它不仅要描述数据类型的数据对象，而且要描述数据对象各元素之间的相互关系。

6. 数据结构的运算

数据结构的主要运算包括：

(1) 建立(Create)一个数据结构；

(2) 消除(Destroy)一个数据结构；

(3) 从一个数据结构中删除(Delete)一个数据元素；

(4) 把一个数据元素插入(Insert)到一个数据结构中；

(5) 对一个数据结构进行访问(Access)；

(6) 对一个数据结构(中的数据元素)进行修改(Modify)；

(7) 对一个数据结构进行排序(Sort)；

(8) 对一个数据结构进行查找(Search)。

1.2 抽象数据类型—— ADT

- 定义：是指基于一个逻辑类型的数据模型以及定义在该模型上的一组操作。每一个操作由它的输入和输出定义。
- 用三元组描述如下：

  （D，S，P）

  其中，D是数据对象，S是D上的关系集，P是对D的基本操作集。
- 抽象的与具体的相对应
- 示例：

int a,b;　则 a和b可以进行+、-、*、/的运算

2和6则是具体的int数据

抽象数据类型的定义

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：<基本操作的定义>

} ADT抽象数据类型名

- 数据对象和数据关系的定义用伪码描述
- 基本操作的定义格式：

  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>

1.3 算法和算法分析

1. 算法 p13

定义：指一系列确定的而且是在有限步能完成的操作。

算法的描述形式有多种，例如可以采用自然语言、数据流程图、伪代码、数学语言或约定的符号来描述，也可以采用高级程序设计语言来描述。

例：求两个正整数 m，n 中的最大数MAX的算法

（1）若 m > n 则 max=m
（2）若 m <= n 则 max=n

算法的重要特性 P13

(1) 有穷性：能在执行有穷步后结束

(2) 确定性：每条指令有确切的含义，对于相同的输入执行相同的路径，得到相同的输出

(3) 输入：**0**至多个输入

(4) 输出：**1**至多个输出

(5) 有效性**(**可行性**)**（用于描述算法的操作都是足够基本的）

- 问题：程序是不是算法？

描述算法（用类C语言）的书写规则

- 所有算法均以函数形式给出，算法的输入数据来自参数表；
- 参数表的参数在算法之前均进行类型说明；
- 有关结点结构的类型定义,以及全局变量的说明等均在算法之前进行说明。

2. 算法与数据结构的关系

- 计算机科学家沃斯（N.Wirth）提出的:

  “算法+数据结构=程序”

  揭示了程序设计的本质：对实际问题选择一种好的数据结构，加上设计一个好的算法，而好的算法很大程度上取决于描述实际问题的数据结构。算法与数据结构是互相依赖、互相联系的。

- 一个算法总是建立在一定数据结构上的；反之，算法不确定，就无法决定如何构造数据。

算法与数据结构关系举例

例1：编写程序查询某城市某人的电话号码

建立一张登记表，存放2个数据项：

姓名+Tel

好的算法取决于这张表的结构及存储方式：

- 将表中结点按照姓名顺序地存储在计算机中，依次查找，可能遍历整个表都找不到。
- 建立一张姓氏索引表：姓+表中的起始地址

则不需查找其他姓氏，查找效率得到提高。

算法与数据结构关系举例

例2:

设计一个考试日程安排表，使在尽可能短的时间内安排完考试，要求同一个学生选修的几门课程不能安排在同一个时间内。

姓名	选修1	选修2	选修3
	A	B	E
	C	D	
	C	E	F
	D	F	A
	B	F	

算法与数据结构关系举例

- 解决该问题，首先选择一个合适的数据结构。用无向图表示，图中的顶点表示课程，不能同时考试的课程之间连上一条边。则该问题就抽象成对该无向图进行“着色”操作，即用尽可能少的颜色去给图中每个顶点着色，使得任意两个相邻的顶点着不同的颜色。同一种颜色表示一个考试 时间。

- 答案：1：A,C 2:B,D 3:E 4: F

- 解决问题的关键步骤是先选取合适的数据结构表示问题，才能写出有效的算法。

3. 算法设计的要求 p13~p14

（1）正确性

四层含义：

a. 程序不含语法错误

b. 程序对于几组输入数据能够得出满足规格说明要求的结果

c. 程序对于精心选择的典型、苛刻而带有刁难性的几组输入数据能够得出满足规格说明要求的结果

d.程序对于一切合法的输入数据都能产生满足规格说明要求的结果

3. 算法设计的要求 p13~p14

（2）可读性

首先是给人读，然后才是机器执行

（3）健壮性　容错性

当输入数据非法时，算法仍能做出反应或进行处理

（4）效率与低存储量需求

算法的效率

- 定义：一个算法如果能在所要求的**资源限制**内将问题解决好，则称这个算法是**有效率的**。

  例如，一个资源限制是：可用来存储数据的全部空间——可能是分离的内存空间限制和磁盘空间限制以及允许执行每一个子任务所需要的时间。

4．算法的分析——算法性能的评价

- 评价标准：

  1）算法所需的计算时间

  2）算法所需的存储空间

  3）算法的简单性

- 度量算法执行时间的两种方法　p14

  1）事后统计法

  此方法有两个缺陷：　p14

  (1)必须先运行依据算法编制的程序；

  (2)所得时间的统计量依赖于计算机的硬件、软件等环境因素，有时容易掩盖算法本身的优劣。

4．算法的分析——算法性能的评价

- 2）事前分析估算法

  此方法取决于多个因素：p14

  a. 依据的算法选用何种策略

  b. 问题的规模

  c. 书写程序的语言，一般来说，实现语言的级别越高，执行效率就越低

  d.编译程序所产生的机器代码的质量

  e.机器执行指令的速度

5.时间复杂度

（1）定义：

一般情况下，算法中*基本操作重复执行*的时间是问题规模n的某个函数f(n)，算法的时间量度记作

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

它表示随问题规模n的增大，算法执行时间的增长率和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**。

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];  //基本操作
        }
```

由于是一个三重循环，每个循环从1到n，则总次数为: $n\times n\times n=n^3$，时间复杂度为$T(n)=O(n^3)$

5 . 时间复杂度

- 语句的**频度**：该语句重复执行的次数。频度统计法

引例:

（a） {++x; s=0；}

++x 的频度为1 时间复杂度为O(1)

（b） for (i=1; i<=n; ++i) {++x; s+=x;}

++x的频度为n 时间复杂度为O(n)

（c） for (j=1;j<=n;++j)

for (k=1;k<=n;++k) {++x; s+=x;}

++x的频度为n^2 时间复杂度为$O(n^2)$

- 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$A(n)=O(n^m)$　证略。

（d） **for(i=2; i<=n; ++i)**

for(j=2; j<=i-1; ++j)

{++x; a[i] [j]=x;}

- 语句频度为：

$1+2+3+\ldots+n-2=(1+n-2)\times(n-2)/2$

$=(n-1)(n-2)/2$

$=n^2-3n+2$

∴时间复杂度为$O(n^2)$

算法的时间复杂度有多种表现形式，常见的有：

(1)常量时间（Constant-time algorithm）：O(1)；

(2)线性时间（Linear-time algorithm）：O(n)；

(3)平方时间（Quadratic-time algorithm）：$O(n^2)$；

(4)立方时间（Cubic-time algorithm）：$O(n^3)$；

(5)对数时间（Logarithmic-time algorithm）：O（log n）；

(6)指数时间（Exponential-time algorithm）：$O(a^n)$。

- 以下六种计算算法时间的多项式是最常用的。其关系为：

  $O(1)<O(\log n)<O(n)<O(n\log n)<O(n^2)<O(n^3)$

- 指数时间的关系为：

  $O(2^n)<O(n!)<O(n^n)$

- 当n取得很大时，指数时间算法和多项式时间算法在所需时间上非常悬殊。因此，只要有人能将现有指数时间算法中的任何一个算法化简为多项式时间算法，那就取得了一个伟大的成就。
- 不同数量级时间复杂度的性状图示（p16）

表：时间复杂度和算法运行时间的关系

T(n) / n	O(logn)	O(n)	O(nlogn)	$O(n^2)$	$O(n^3)$	$O(n^5)$	$O(2^n)$	O(n!))
20	4.3us	20us	86.4us	400us	8ms	3.2s	1.05s	771世纪
40	5.3us	40us	213us	1600us	64ms	1.7min	12.7天	2.59*1032世纪
60	5.9us	60us	354us	3600us	216ms	13min	366世纪	2.64*1066世纪

- 有的情况下，算法中基本操作重复执行的次数还随问题的输入数据集不同而不同。例如：

```
void  bubble-sort(int a[ ]，int n)
   for(i=n-1; change=TURE; i>1 && change; --i)
      {
            change=false;
            for(j=0; j<i; ++j)
                 if (a[j]>a[j+1])
                      { a[j] ←→a[j+1];//基本操作：交换序列中相邻两个基本点整数
                           change=TURE；
                      }
```

- 最好情况（从小至大排列）：0次
- 最坏情况（从大至小排列）：1+2+3+…+n-1

  $=n(n-1)/2$
- 平均时间复杂度为: $O(n^2)$

结论

（1）当f(n)为对数函数、幂函数、或它们的乘积时，算法的运行时间是可以接受的，称这些算法是有效算法；当f(n)为指数函数或阶乘函数时，算法的运行时间是不可接受的，称这些算法是无效的算法。

（2）随着n值的增大，增长速度各不相同，n足够大时，存在下列关系：

对数函数<幂函数<指数函数

常见函数的增长率

O(1)	常量阶，与n无关
O(log n)	log n阶
O(n)	n阶
O(n log n)	n log n阶
$O(n^2)$	平方阶
$O(n^3)$	立方阶
$O(2^n)$	指数阶

- 增长率由慢到快 图示见p16图1-7
- 尽量少用指数阶的算法
- 说明：教材p17第2行 除特别指明外，均指最坏情况下的时间复杂度

6. 空间复杂度

（1）存储算法本身所占用的空间

（2）算法的输入/输出数据占用的空间

（3）算法在运行过程中临时占用的辅助空间

- 原地工作：若辅助空间相对于输入数据量是常数，则称此算法是原地工作。
- 若所占空间量依赖于特定的输入，按最坏情况来分析。