



# 数 据 结 构

---

## 第一章 绪论 Introduction

计算机学院  
霍红卫








# 第一章 绪论



- 1.1 什么是数据结构
- 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
  - 1.4.1 算法
  - 1.4.2 算法设计的要求
  - 1.4.3 算法效率的度量
  - 1.4.4 算法的存储空间的需求



# 1.1 什么是数据结构

## 例1 图书馆的书目检索系统自动化问题

<b>001</b>	高等数学	樊映川	<b>S01</b>	...
<b>002</b>	理论力学	罗远祥	<b>L01</b>	...
<b>003</b>	高等数学	华罗庚	<b>S01</b>	...
<b>004</b>	线性代数	滦如书	<b>S02</b>	...
				

高等数学	<b>001,003, ...</b>
理论力学	<b>002, ...</b>
线性代数	<b>004, ...</b>
	

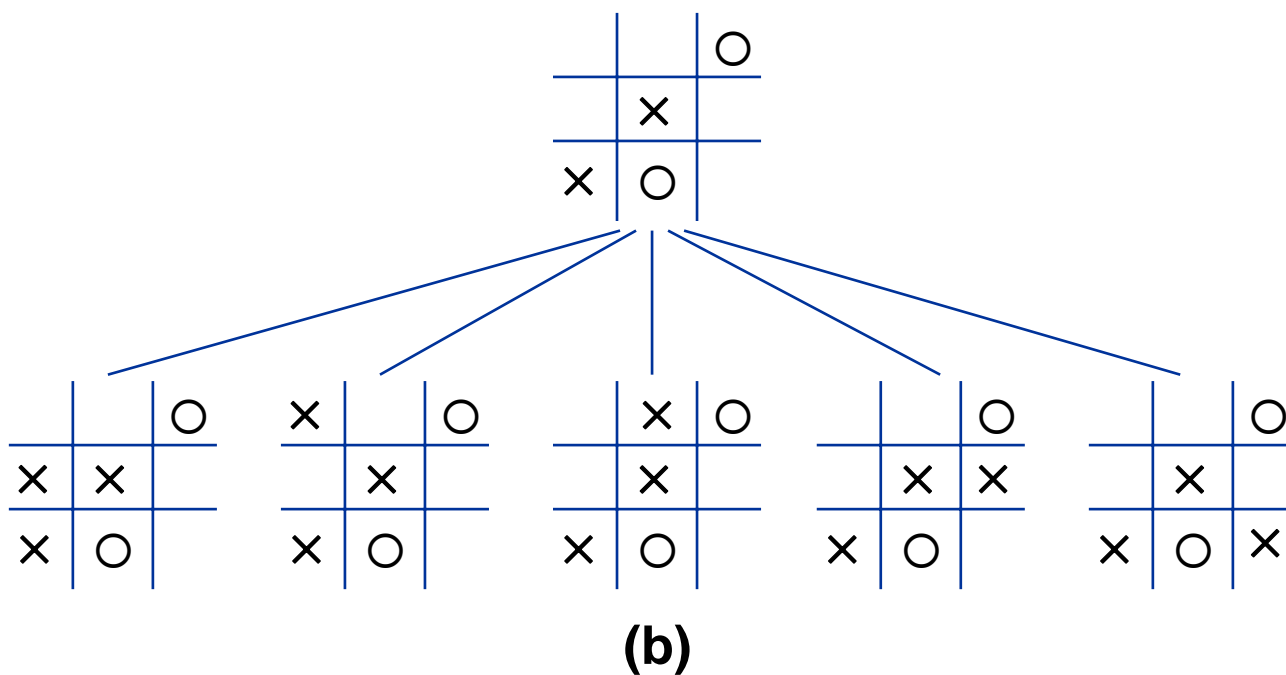
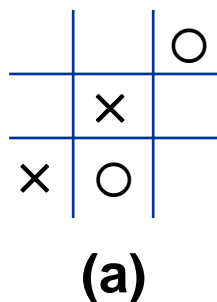
樊映川	<b>001, ...</b>
华罗庚	<b>003, ...</b>
滦如书	<b>004, ...</b>
	

<b>L</b>	<b>002, ...</b>
<b>S</b>	<b>001,003, ...</b>
	

图书目录文件示例

# 1.1 什么是数据结构

## 例2 计算机和人对弈问题



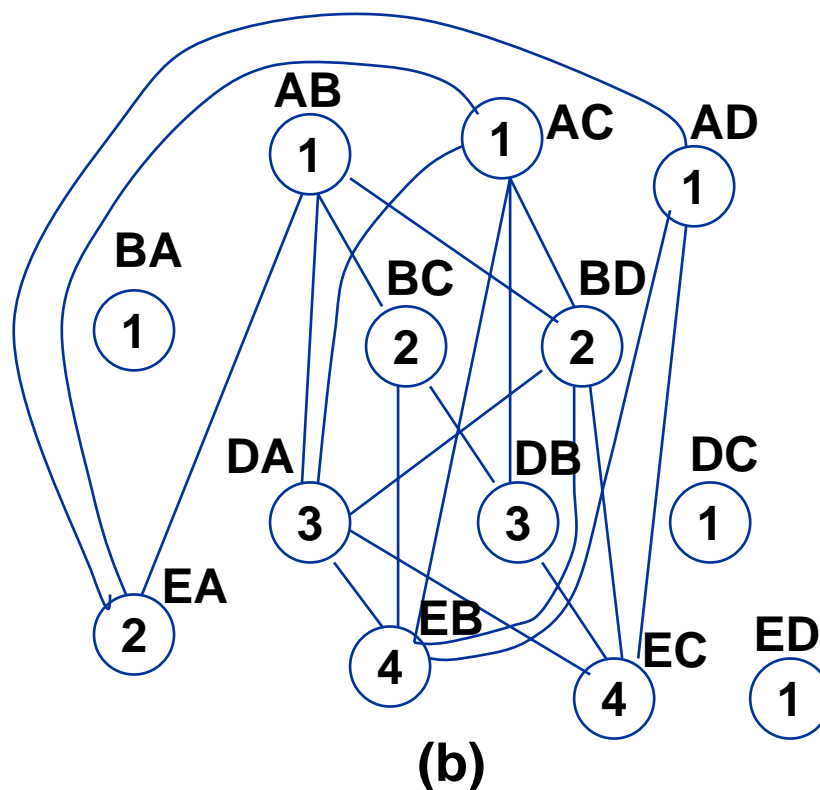
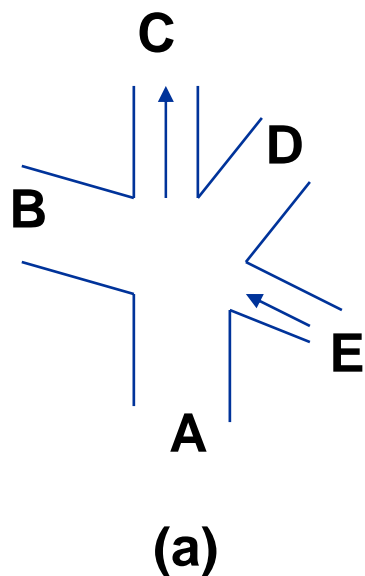
井字棋 对弈树

(a)井字棋 对弈树

(b) 对弈树的局部

# 1.1 什么是数据结构

## 例3 多叉路口交通灯的管理问题



五叉路口交通管理示意图

(a) 五叉路口

(b) 表示通路的图



## 1.2 基本概念和术语

- **数据 (Data)**: 是对信息的一种符号表示。在计算机科学中是指所有能输入到计算机中并被计算机程序处理的符号的总称。
- **数据元素 (Data Element)**: 是数据的基本单位, 在计算机程序中通常作为一个整体进行考虑和处理。
- 一个数据元素可由若干个数据项组成。数据项是数据的不可分割的最小单位。
- **数据对象 (Data Object)**: 是性质相同的数据元素的集合。是数据的一个子集。
- **数据结构 (Data Structure)**: 是相互之间存在一种或多种特定关系的数据元素的集合。数据结构的形式定义为: 数据结构是一个二元组:

$$\text{Data-Structure} = (D, S)$$

其中: D是数据元素的有限集, S是D上关系的有限集。



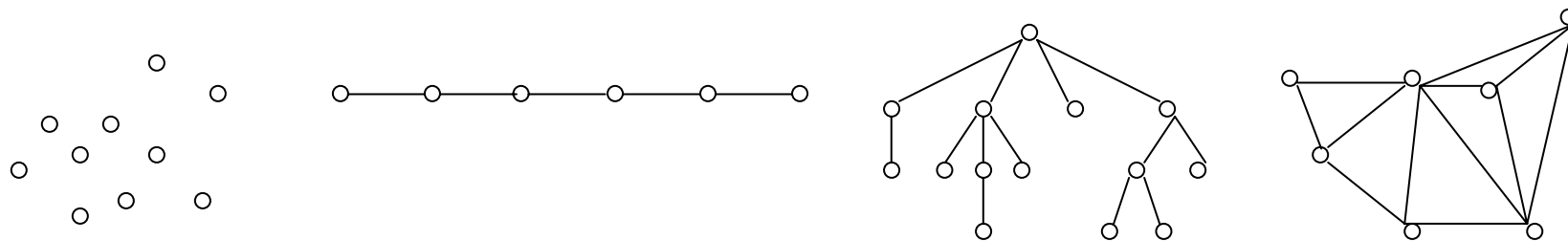
## 1.2 基本概念和术语

- 数据之间的相互关系称为逻辑结构。
- 数据的逻辑结构是独立于计算机的，对数据元素之间的逻辑关系的描述。从集合的观点，它们可以形式地用一个二元组  $B = (D, R)$  表示，其中  $D$  是数据元素的集合， $R$  是  $D$  上关系的集合。
- 按集合的观点，数据的逻辑结构有两个要素：一是数据元素；二是关系。

## 1.2 基本概念和术语

通常分为四类基本结构：

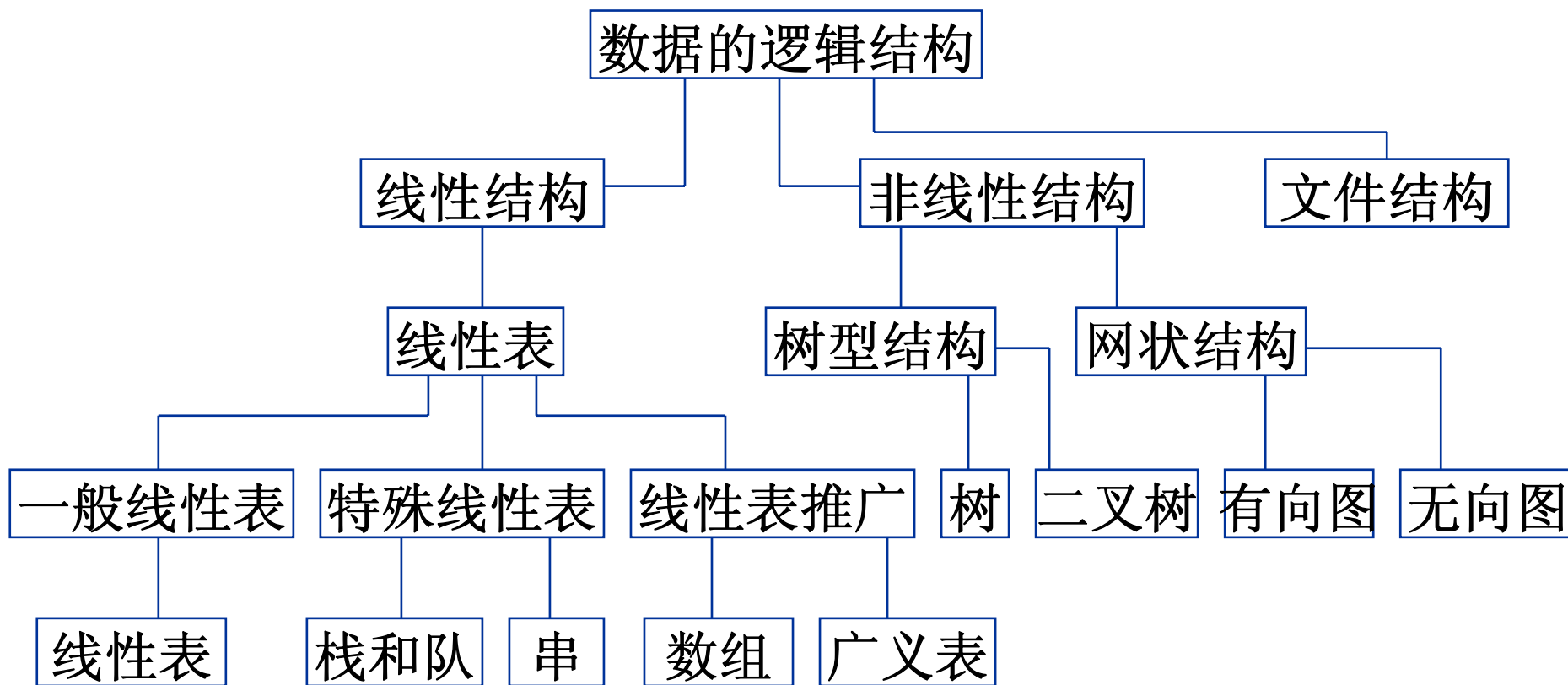
- 集合 结构中的数据元素除了同属于一种类型外，别无其它关系。
- 线性结构 结构中的数据元素之间存在一对一的关系。
- 树型结构 结构中的数据元素之间存在一对多的关系。
- 图状结构或网状结构 结构中的数据元素之间存在多对多的关系。





## 1.2 基本概念和术语

数据的逻辑结构按关系分为线性结构（关系是线性的）和非线性结构（关系是非线性的）。





## 1.2 基本概念和术语

- 数据的物理结构
  - 是指数据的逻辑结构在计算机中的映象，即存储表示。映象包括数据元素的映象和数据关系的映象。数据元素的映象是结点，即在计算机内用一结点表示一个数据元素（结点是数据结构讨论的基本单位）。关系的映象有两种，顺序映象和非顺序映象。



## 1.2 基本概念和术语

- 数据结构主要指逻辑结构和物理结构
- 顺序存储结构与非顺序存储结构
  - 顺序存储机构是逻辑上相邻的数据元素存储的物理位置上相邻的存储单元里，元素的关系由存储单元的邻接关系来体现。
  - 非顺序存储结构是数据元素可以在计算机内任意位置上存放（它不要求逻辑上相邻的元素在物理位置上也相邻），它们的逻辑关系用指针来链接。所以非顺序存储结构又叫链式存储结构。链式存储结构将数据元素存放的存储单元分为两个部分，分别存放数据和指针，称为数据域和指针域。



## 1.4 算法和算法分析

- **算法**：是对特定问题求解步骤的一种描述。算法是指令的有限序列，其中每一条指令表示一个或多个操作。算法具有以下五个特性：
  - **有穷性** 一个算法必须总是在执行有穷步之后结束，且每一步都在有穷时间内完成。
  - **确定性** 算法中每一条指令必须有确切的含义。不存在二义性。且算法只有一个入口和一个出口。
  - **可行性** 一个算法是可行的。即算法描述的操作都是可以通过已经实现的基本运算执行有限次来实现的。
  - **输入** 一个算法有零个或多个输入，这些输入取自于某个特定的对象集合。
  - **输出** 一个算法有一个或多个输出，这些输出是同输入有着某些特定关系的量。



## 1.4 算法和算法分析

- 算法评价标准

- 正确性——合理的数据输入下，在有限的运行时间内得出正确的结果。
- 健壮性——对不合理的数据输入的反应和处理能力。
- 可读性——供人们阅读的方便程度。
- 简单性——采用数据结构和方法的简单程度。
- 时间复杂度(计算复杂度)——算法运行时间的相对量度。
- 空间复杂度——算法运行中临时占用空间大小的量度。



## 1.4 算法和算法分析

- 时间复杂度(**Time Complexity**)

- 定义：如果存在两个正常数 $c$ 和 $n_0$ ，对于所有的  $n \geq n_0$ ，有  $|f(n)| \leq c |g(n)|$ ，则记作  **$f(n)=O(g(n))$**

例 1 **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]; }**

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 $T(n)=O(n^3)$

**频度：**是指该语句重复执行的次数



## 1.4 算法和算法分析

例 2  $\{++x; s=0;\}$

- 将x自增看成是基本操作，则语句频度为 1，即时间复杂度为  $O(1)$
- 如果将s=0也看成是基本操作，则语句频度为 2，其时间复杂度仍为  $O(1)$ ，即常量阶。

例 3 **for**(**i=1; i<=n; ++i**)

**{ ++x; s+=x; }**

- 语句频度为： $2n$  其时间复杂度为： $O(n)$
- 即时间复杂度为线性阶。



## 1.4 算法和算法分析

例 4 `for(i=1;i<=n;++i)`  
    `for(j=1;j<=n;++j)`  
        `{++x;s+=x;}`

- 语句频度为：  $2n^2$
- 其时间复杂度为：  $O(n^2)$ ，即时间复杂度为平方阶。

定理：若  $A(n) = a_m n^m + a_{m-1} n^{m-1} + \dots + a_1 n + a_0$  是一个  $m$  次多项式，则

$$A(n) = O(n^m)$$

证略。





## 1.4 算法和算法分析

- 例 5 

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

语句频度为:

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

$\therefore$  时间复杂度为  $O(n^2)$

- 一个算法时间为  $O(1)$  的算法，它的基本运算执行的次数是固定的。因此，总的时间由一个常数（即零次多项式）来限界。而一个时间为  $O(n^2)$  的算法则由一个二次多项式来限界。



## 1.4 算法和算法分析

- 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。



## 1.4 算法和算法分析

- 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];
                change = TURE; }
    }
```



## 1.4 算法和算法分析

- 最好情况: 0次
- 最坏情况:  $1+2+3+\dots+n-1=n(n-1)/2$
- 平均时间复杂度为:  $O(n^2)$



## 1.4 算法和算法分析

- 空间复杂度 (**Space Complexity**): 算法所需存储空间的度量, 记作:

$$S(n) = O(f(n))$$

其中  $n$  为问题的规模 (或大小)

- 算法运行中占用空间包括算法本身占用, 输入输出数据占用和运行中的临时占用。
- 同一个问题, 算法不同, 运行中的临时空间不同, 即空间复杂性不同。
- 只考虑在运行中为局部变量分配的存贮空间的大小, 一般也以数量级表示。