

# 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

- 数据结构习题集（C语言版）

严蔚敏，吴伟民，米宁编著 清华大学出版社





## 参考教材

---

- Data Structures, Algorithms, and applications in C++ , Sartaj Sahni, 机械工业出版社
- 数据结构与算法分析---- C语言描述, Mark Allen Weiss著, 机械工业出版社



# 课时安排

- 56学时授课+12学时上机
- 12学时上机=4学时/次\*3次
- 3次上机的时间：第4、7、8周六上午，西部校区综合楼B601
- 课件下载：  
[dlut2014data@163.com](mailto:dlut2014data@163.com), 信箱密码：2014algorithm





# 课程期末总成绩计算

---

1. 期末试卷 ---- 80%
2. 平时作业+平时测验+上机 ---- 20%

# 联系方式

- Email: [xhlin@dlut.edu.cn](mailto:xhlin@dlut.edu.cn)
- 办公地点: 创新园大厦A座920
- 办公室电话: 84706002-3920

欢迎大家课程答疑、算法探讨和交流



# 第一章 绪论

- 本章主要内容:
  1. 什么是**数据结构**?
  2. 什么是**算法**?
  3. 该课程包含**哪些内容**?



# 1.1 什么是数据结构

计算机求解问题的过程:

问题

分析

数学模型

算法

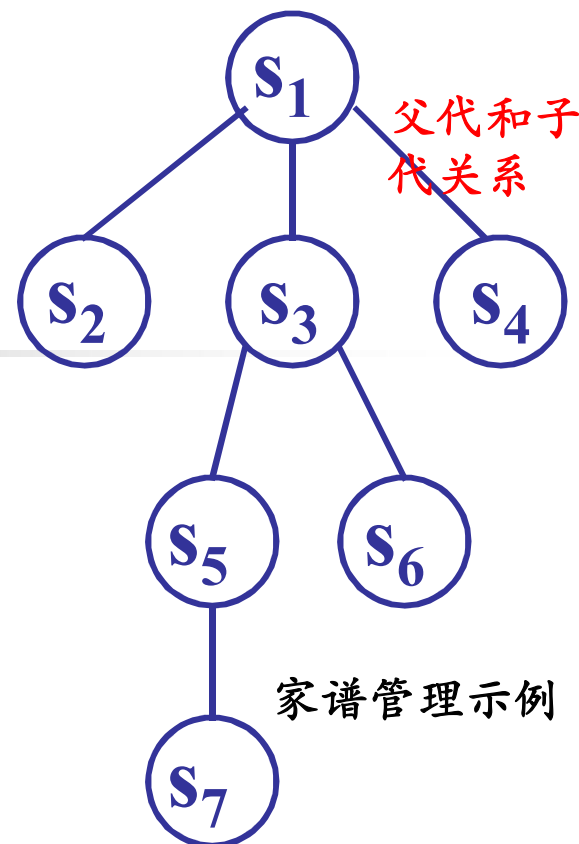
编码  
问题  
求解

操作对象，关系  
用数学语言加以  
描述

操作

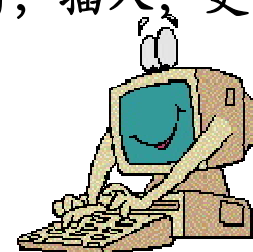
**数据结构:问题的数学模型**

- 操作对象称为 **数据元素**
- 操作对象之间的关系称为 **结构** (逻辑结构)



Family=(D,S),  $D=\{s_i:1\leq i\leq 7\}$ ,  
 $S=\{<s_1,s_2>, <s_1,s_3>, <s_1,s_4>,<s_3,s_5>, <s_3,s_6>, <s_5,s_7>\}$

操作: 查询, 插入, 更新





## 1.1 什么是数据结构

---

- **数据结构**: 问题的数学模型, 是指互相之间存在着一种或多种**特定关系**的数据元素的集合
- **算法**: 求解问题的策略, 操作步骤

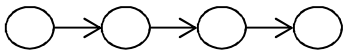


## 1.2 基本概念和术语

- **数据**：客观事物的符号表示，计算机加工处理的对象，所有能**输入**到计算机中并能被计算机**处理**的符号总称。
- **数据元素**：即操作对象，是数据的基本单位。在不同的问题中，数据元素又可称为**元素、结点、顶点、记录**等。
- **数据项**：一个数据元素可由若干个数据项。例如每个家庭成员包含**姓名、性别、出生日期**…等数据项
- **数据对象**：性质相同的数据元素的集合，是数据的一个子集。

## 1.2 基本概念和术语

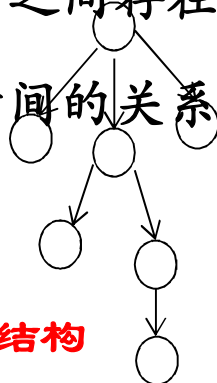
- 操作对象之间的**关系**称为**结构**（**逻辑结构**）
- **逻辑结构**：描述元素之间的逻辑关系，而不考虑具体存储，分为线性结构和非线性结构（**树形结构**，**图状结构**，**集合结构**）
- 逻辑结构的种类：**我们只研究前3种**

1. **线性结构** 数据元素之间存在着**一对一**的关系 

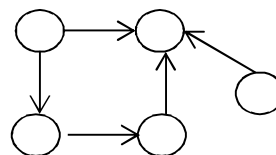
2. **树形结构** 数据元素之间存在着**一对多**的关系

3. **图状结构** 数据元素之间存在着**多对多**的关系，也称网状结构

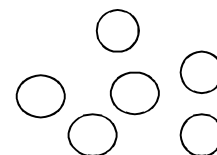
4. **集合结构** 数据元素间的关系是“**属于同一个集合**”



**树形结构**

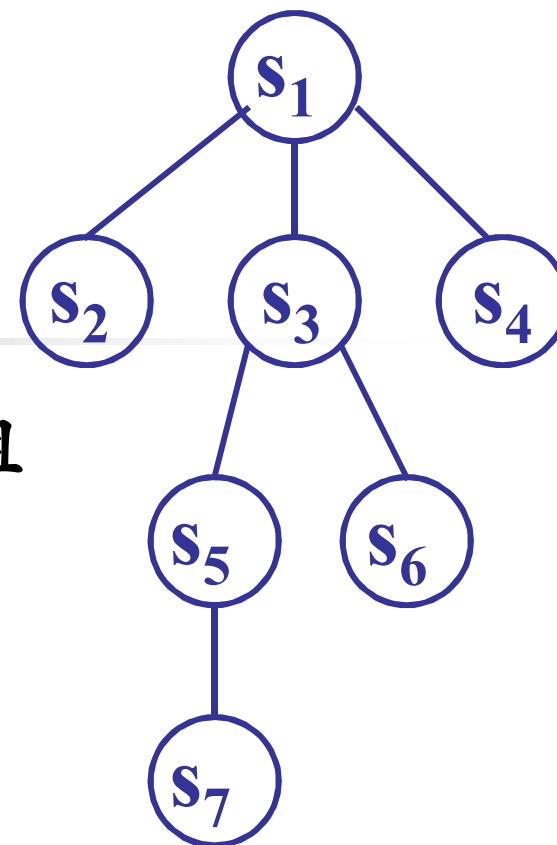


**图状结构**



**集合结构**

## 1.2 基本概念和术语



□ 数据结构形式定义：采用二元组

**Data\_structure**=(**D**, **S**)

1. Data\_structure是为该数据结构起的名字
2. D代表该数据结构所有操作对象的集合
3. S代表所有操作对象关系的集合

□ 例如家谱管理：Family=(D,S)，其中

**D** = { $s_i: 1 \leq i \leq 7$ } ,

**S** = { $\langle s_1, s_2 \rangle, \langle s_1, s_3 \rangle, \langle s_1, s_4 \rangle, \langle s_3, s_5 \rangle, \langle s_3, s_6 \rangle, \langle s_5, s_7 \rangle$ }

数据结构定义的2个要素：数据元素的集合**D**，关系的集合**S**

## 1.2 基本概念和术语

- **物理（存储）结构：** 数据结构在计算机中的表示
  - 设计数据结构的存储结构时要存放所有数据元素的值和它们之间的逻辑关系
- 2种存储结构：
  1. 顺序存储映像——**顺序存储结构**借助数据元素在存储器中的相对位置来表示数据元素之间的逻辑关系
  2. 非顺序存储映像——**链式存储结构**借助指示数据元素存储地址的指针来表示数据元素之间的逻辑关系

# 顺序存储结构—示例

排队等待招聘面试：排队的人 $a_1, a_2, a_3, a_4, \dots$

	姓名	性别	年龄	学历	专业		存储空间
$a_1$	王晓林	女	22	本科	计算机科学与技术	A0	
$a_2$	陈红	女	25	研究生	自动化	A0+L	
$a_3$	刘建平	男	23	本科	电子工程	A0+2L	
$a_4$	张立立	男	26	研究生	数学	A0+3L	
	...	...	...	...	...	...	...

顺序存储：找一块连续的存储空间，从队列的第一个数据元素开始存放，该空间第一个位置存王晓林的信息，第二个位置存陈红的信息，依次存下去…存放位置体现逻辑关系，存放顺序和排队顺序一致

# 顺序存储结构—示例

排队等待招聘面试：排队的人 $a_1, a_2, a_3, a_4, \dots$

	姓名	性别	年龄	学历	专业		存储空间
$a_1$	王晓林	女	22	本科	计算机科学与技术	A0	王晓林, 女, ...
$a_2$	陈红	女	25	研究生	自动化	A0+L	陈红, 女, ...
$a_3$	刘建平	男	23	本科	电子工程	A0+2L	刘建平, 男, ...
$a_4$	张立立	男	26	研究生	数学	A0+3L	张立立, 男, ...
	...	...	...	...	...	...	...

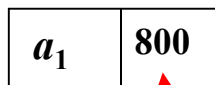
顺序存储: 找一块连续的存储空间, 从队列的第一个数据元素开始存放, 该空间第一个位置存王晓林的信息, 第二个位置存陈红的信息, 依次存下去... 存放位置体现逻辑关系, 存放顺序和排队顺序一致

# 链式存储结构—示例

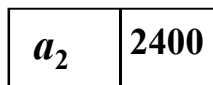
- 为数据结构中的每个数据元素分配**一块存储空间**（**称为结点**），存放数据元素的**值**和**逻辑关系**。
  - 结点中存放逻辑关系的那部分空间称为“指针”
  - 链式存储结构每个结点包含二部分：**值和指针**
  - **指针**存放的是和该数据元素有特定逻辑关系的数据元素的地址
- 排队等待招聘面试：排队的人 $a_1, a_2, a_3, a_4, \dots$

**h=1000**

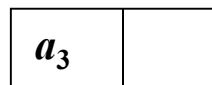
1000



800

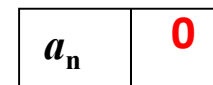


2400



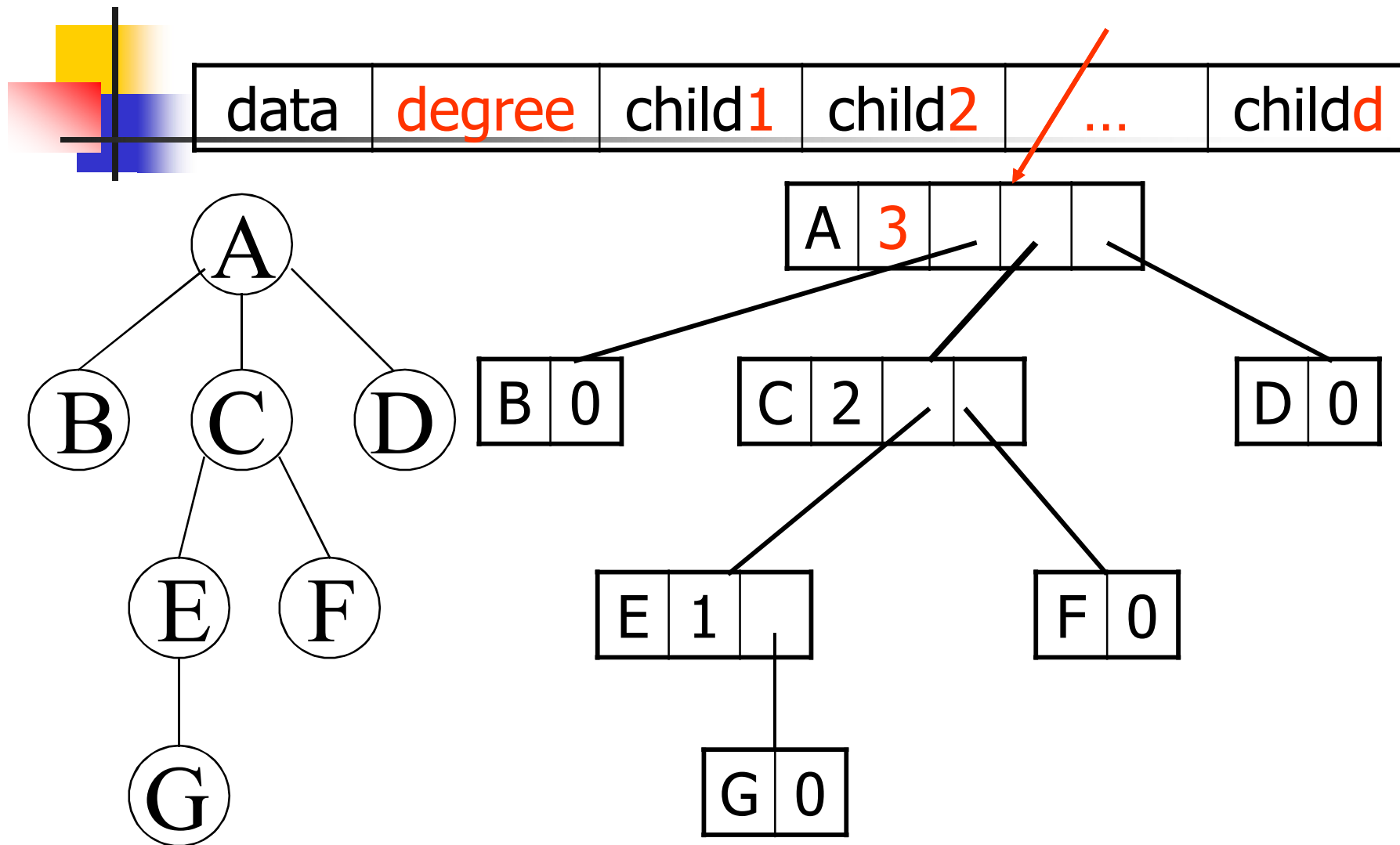
.....

1600



指针：存放与 $a_1$ 有逻辑**关系** $a_2$ 的地址

# 链式存储结构—示例







# 逻辑结构与物理结构的关系

- 数据的**逻辑结构**在计算机**存储里的实现**
- 一种逻辑结构可以采用**不同**的存储结构
- 运算定义在逻辑结构之上，如插入、删除运算的定义并不考虑数据的存储结构
- 运算的实现依赖于具体的存储结构

□ 本学期主要介绍线性、树、图结构的定义以及它们的存储结构，相应基本操作（算法）的实现



## 1.3 抽象数据类型的表示与实现

- 数据结构采用程序设计语言中的**数据类型**实现
  - 数据类型：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：int, char, float, double等
  - 简单的数据结构可用程序设计语言提供的数据类型实现；复杂的数据结构需要自定义数据类型。

## 1.3 抽象数据类型的表示与实现

- **抽象数据类型** (Abstract Data Type 简称ADT): 是指一个数学模型以及定义在此数学模型上的一组操作。
- 抽象数据类型通常用  $(D, R, P)$  三元组表示:  
D是数据对象, R是D上的关系的集合, P是D上的操作的集合
- 形式化定义格式:

### **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: 〈基本操作的定义〉  
} **ADT** 抽象数据类型名



# 定义格式

- 基本操作的定义格式为:

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〉

- 参数2种

1. 赋值参数 -- 只为操作提供输入值。
  2. 引用参数 -- 以"&"打头, 除可提供输入值外, 还将返回操作结果。
- 初始条件: 描述了操作执行之前数据结构和参数应满足的条件, 若不满足, 则操作失败, 并返回相应出错信息。
  - 操作结果: 说明了操作正常完成之后, 数据结构的变化状况和应返回的结果。
  - 若初始条件为空, 则省略之。

## ADT Complex

{ 数据对象:  $D = \{e1, e2 \mid e1, e2 \in \text{RealSet}\}$

数据关系:  $R1 = \{(e1, e2) \mid e1 \text{ 是实数部分}, e2 \text{ 是复数的虚数部分}\}$

基本操作:

**AssignComplex**( &Z, v1, v2 )

操作结果: 构造复数 Z, 其实部和虚部 分别被赋以参数 v1 和 v2 的值。

**DestroyComplex**( &Z )

操作结果: 复数 Z 被销毁。

**GetReal**( Z, &realPart )

初始条件: 复数已存在。

操作结果: 用 realPart 返回复数 Z 的实部值。

**GetImag**( Z, &ImagPart )

初始条件: 复数已存在。

操作结果: 用 ImagPart 返回复数 Z 的虚部值。

**Add**( z1, z2, &sum )

初始条件: z1, z2 是复数。

操作结果: 用 sum 返回两个复数 z1, z2 的 和值。

} **ADT Complex**

- 假设: z1 和 z2 是上述定义的复数, 则 **Add(z1, z2, z3)** 操作的结果即为  $z3 = z1 + z2$



# 抽象数据类型的表示与实现

- 抽象数据类型需要通过固有数据类型(高级编程语言中已实现的数据类型)来实现。

- 存储结构的定义

```
typedef struct {  
    float e1;  
    float e2;  
}complex;
```

- 基本操作的实现

```
void add( complex z1, complex z2, complex &sum ) {  
    // 以 sum 返回两个复数 z1, z2 的和  
    sum.e1 = z1.e1 + z2.e1;  
    sum.e2 = z1.e2 + z2.e2;  
}
```

所有的基本操作均要编码实现



## 1.4 算法和算法分析

- 算法--求解特定问题的**有限**的操作步骤
- 算法的5个特性：
  1. 有限性：不能无限制的操作下去，不能死循环
  2. 确定性：每一步必须有明确含义，不能有歧义
  3. 可行性：每一步利用现有技术是可以实现的
  4. 输入：0个或多个输入
  5. 输出：1个或多个输出



## 设计算法求全班 $n$ 个学生的平均成绩

1. 输入 $n$ ;
2.  $sum=0$ ;
3.  $i=0$ ;
4. 若 $i==n$ 则转8, 否则
5.  $i++$ ; 读入第 $i$ 个学生的成绩到变量 $x$ ;
6.  $sum=sum+x$ ;
7. 转4;
8.  $ave=sum/n$ ;
9. 输出 $ave$ ;





## 设计算法求全班 $n$ 个学生的平均成绩

1. `Scanf("%d",& $n$ );`
2.  `$sum$ =0;`
3.  `$i$ =0;`
4. `While( $i$ !=  $n$ ){`
5.  `$i$ ++; Scanf("%d",&  $x$ );`
6.  `$sum$ =  $sum$ +  $x$  ;`
7. `}`
8.  `$ave$ = $sum$ / $n$ ;`
9. `printf("%d",  $ave$ ) ;`

## 设计算法求全班 $n$ 个学生的平均成绩

```
void average() {
```

```
    Scanf("%d",& $n$ );
```

```
     $sum$ =0;
```

```
     $i$ =0;
```

```
    While( $i$ !=  $n$ ){
```

```
         $i$ ++; Scanf("%d",&  $x$ );
```

```
         $sum$ =  $sum$ +  $x$  ;
```

```
    }
```

```
     $ave$ = $sum$ / $n$ ;
```

```
    printf("%d",  $ave$ ) ;
```

```
}
```

本课程的算法采用类C语言描述  
见1.3节



# 算法的设计要求

---

- 正确性
- 健壮性
- 可读性
- 效率和低存储量需求
  - 效率——算法的执行时间
  - 存储量——算法执行过程中所需要的最大存储空间



# 正确性

- 满足需求
- 对算法是否“正确”的理解可以有以下四个层次：
  1. 程序中不含语法错误
  2. 程序对于几组输入数据能够得出满足要求的结果
  3. 程序对于精心选择的、典型、苛刻且带有刁难性的几组输入数据能够得出满足要求的结果
  4. 程序对于一切合法的输入数据都能得出满足要求的结果

通常以第3层意义的正确性作为衡量一个算法是否合格的标准。



# 算法效率的度量

## □ 2种度量方法:

### 1. 事后统计的方法—编码运行，实际测量

不利于较大范围内的算法比较（异时，异境）

### 2. 事前分析估算的方法

- 算法本身选用的策略
- 问题的规模
- 书写程序的语言
- 编译产生的机器代码质量
- 机器执行指令的速度
- 一个特定算法的运行工作量只依赖于问题的规模

# 事前分析估算方法--度量方式

- 从算法中选取一种对于所研究的问题来说是**基本操作**的**原操作**，以该基本操作重复执行的次数作为算法的时间量度。
- $T(n)=O(f(n))$
- 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，称作算法的渐近时间复杂度，简称时间复杂度。
- 原操作执行次数和包含它的语句的频度相同。语句的频度指的是该语句重复执行的次数。

## 渐进时间复杂性—上界 $O$

- 设 $T, f$ 是定义域为自然数的函数,  $T(n)=O(f(n))$ :  
若存在正数 $c$ 和 $n_0$ , 使得对一切 $n \geq n_0$ ,  $0 \leq T(n) \leq cf(n)$

$$\lim_{n \rightarrow \infty} \frac{T(n)}{f(n)} = c < \infty$$

## 例子----矩阵相乘

```
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]=c[i][j]+a[i][k]*b[k][j];  
  }
```

**$O(n^3)$**



# 例子

语句	频度	时间复杂度
<code>{++x; s=0;}</code>	1	$O(1)$
<code>for (i=1; i&lt;=n; ++i)</code>  <code>{++x; s+=x;}</code>	n	$O(n)$
<code>for (j=1; j&lt;=n; ++j)</code>  <code>for (k=1; k&lt;=n; ++k)</code>  <code>{++x; s+=x;}</code>	$n*n$	$O(n*n)$

## 练习一下

下面程序段的时间复杂度为( )。

```
for(int i=0; i<m; i++)  
    for(int j=0; j<n; j++)  
        a[i][j]=i*j;
```

A.  $O(m^2)$

B.  $O(n^2)$

C.  $O(m*n)$

D.  $O(m+n)$





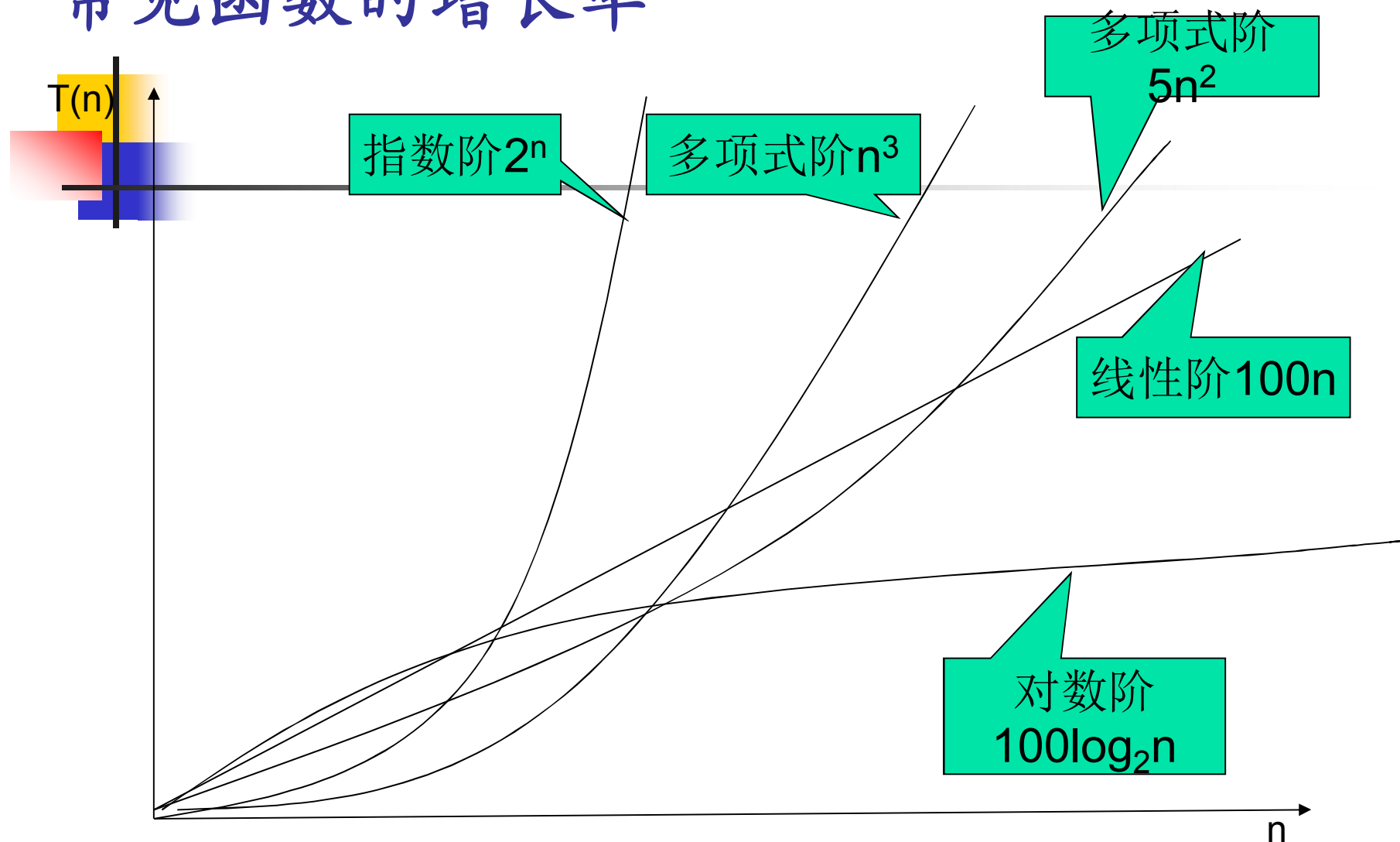
# 说明

---

- $O(1)$  —常量阶
- $O(n)$  —线性阶
- $O(n^2)$  —平方阶
- $O(\log n)$  —对数阶
- $O(2^n)$  —指数阶

多项式阶（ $O(n^k)$ ），不希望用指数阶

# 常见函数的增长率





# 算法的存储空间需求

- 空间复杂度可以作为算法所需存储空间的量度
- $S(n)=O(f(n))$
- 若额外空间相对于输入数据量来说是常数，则称此算法为**原地工作**。
- 如果所占空间量依赖于特定的输入，则除特别指明外，均按**最坏**情况分析。

# 本章学习要点

1. 熟悉各名词、术语的含义，掌握基本概念。
2. 理解算法五个要素的确切含义。
3. 掌握估算算法时间复杂度的方法。

# 学习方法

- 课前预习，上课认真听讲，做好课堂笔记。
- 认真完成作业，多做练习。
- 勤上机实践，掌握课程的核心内容。

