
# 数据结构与算法设计

# 课程内容

课程内容：**数据结构部分**

| | | | |
|---|---|---|---|
| 概述 | 数组与广义表 | 图 | 外部排序 |
| 线性表 | 串 | 查找 | |
| 栈与队列 | 树 | 内部排序 | |

课程内容：**算法设计部分**

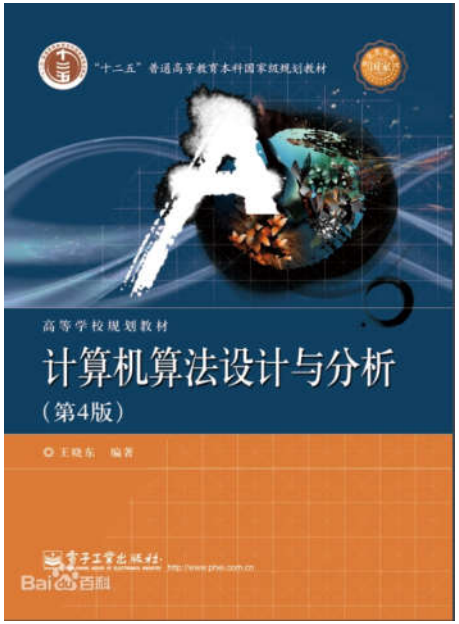
| | | | |
|---|---|---|---|
| 概述 | 贪心 | …… | 计算模型 |
| 分治 | 回溯 | …… | 可计算理论 |
| 动态规划 | …… | …… | 计算复杂性 |

# 教材


数据结构

殷人昆编著

清华大学出版社第二版




计算机算法设计与分析

王晓东编著

电子工业出版社第四版

# 参考教材

| | | |
|---|---|---|
| 数据结构（C语言版） | 严蔚敏 吴伟民编 | 清华大学出版社 |
| 数据结构题集（C语言版） | 严蔚敏　吴伟民 | 清华大学出版社 |
| 数据结构与算法（C++） | 张铭 王腾蛟 赵海燕编 | 高等教育出版社 |
| 算法导论（第三版） | Cormen等著，潘金贵等译 | 机械工业出版社 |
| 计算理论导引 | Sipser著，唐常杰等译 | 机械工业出版社 |
| 算法引论 | Manber著，黄林鹏等译 | 电子工业出版社 |
| 算法艺术与信息学竞赛 | 刘汝佳等 | 清华大学出版社 |

# 课程信息

乐学平时成绩30分+期末考试：70分

## 前置知识：C++中指针和引用的区别



指针是一个实体，需要分配内存空间。引用只是变量的别名，不需要分配内存空间。

引用在定义的时候必须进行初始化，并且不能够改变。
指针在定义的时候不一定要初始化，并且指向的空间可变。

有多级指针，但是没有多级引用，只能有一级引用

# 前置知识：C++中指针和引用的区别

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

举个例子：

交换a和b的值

# 前置知识：C++中指针和引用的区别

① 从现象上看，指针在运行时可改变其所指向的值，而引用一旦和某个对象绑定后就不在改变。

② 从内存上分配看，程序为指针变量分配内存区域，而不用为引用分配内存区域，引用声明时必须初始化，从而指向一个已经存在的对象，引用不能指向空值。

③ 从编译上看，程序在编译时分别将指针和引用添加到符号表上，符号表上记录的是变量名及变量所对应地址。指针变量在符号表上对应的地址值是指针变量的地址值，而引用在符号表上对应的地址值是引用对象的地址值。符号表生成后就不会再改变，指针变量的值可以改，而引用对象不能改。这是使用指针不安全而使用引用安全的主要原因。

④ 从汇编或者机器语言上看，两者没有区别

# 引言


计算机要解决数据如何在如何组织，如何存储，如何高效处理的问题

Contents

# 本章内容

什么是数据结构

基本概念和术语

抽象数据类型

算法和算法分析

# 什么是数据结构

## 什么是数据?

**数据：所有能输入到计算机中，并能被其存储、加工、处理的符号的集合。**

**数据：客观对象的符号表示。**

# 什么是数据结构

数据从计算机处理的对象上看，包含了数据的形式、内容以及组织方式

数据结构 + 数据内容 + 数据流

数据结构指某一数据元素集合中数据元素之间的关系。

~~数据内容指这些数据元素的具体涵义和内容。~~

~~数据流指这些数据元素在系统处理过程中是如何传递和变换的。~~

因此，讨论数据结构时,不讨论数据内容和数据流。

# 什么是数据结构

例1：已知游泳池的长len和宽width，求面积area

已知：游泳池的长len，宽width，

求解：面积area

对象之间的关系：area=len×width

```
int  main ( ) {
        int len, width,area ;
        scanf ( “%d%d”, &len, &width);
        printf ( “area=%d\n”, len*width);
        return 0;
  }
```

# 什么是数据结构

例2：已知学生选课情况如下，请安排课程考试的日程，要求在尽可能短的时间内完成考试，假定一个学生一天只能安排一次考试。

| 姓名 | 选修课1 | 选修课2 | 选修课3 |
|---|---|---|---|
| 刘德驰 | 算法分析（A） | 形式语言（B） | 计算机网络（E） |
| 周星华 | 计算机图形学（C） | 模式识别 （D） | |
| 李连龙 | 计算机图形学（C） | 计算机网络（E） | 人工智能（F） |
| 成　杰 | 模式识别（D ） | 人工智能（F） | 算法分析（A） |
| 令狐盈 | 形式语言（B） | 人工智能（F） | |

# 什么是数据结构

| 姓名 | 选修课1 | 选修课2 | 选修课3 |
| --- | --- | --- | --- |
| 刘德驰 | 算法分析（A） | 形式语言（B） | 计算机网络（E） |
| 周星华 | 计算机图形学（C） | 模式识别 （D） | |
| 李连龙 | 计算机图形学（C） | 计算机网络（E） | 人工智能（F） |
| 成　杰 | 模式识别（D ） | 人工智能（F） | 算法分析（A） |
| 令狐盈 | 形式语言（B） | 人工智能（F） | |

问题涉及的对象：学生——课程

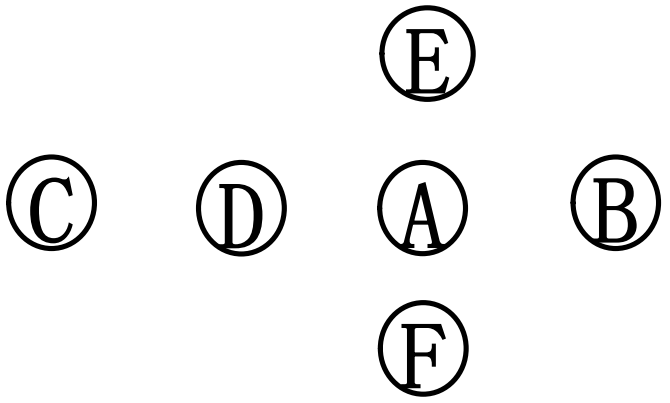
课程之间的关系：同一学生选修的课程之间有关联关系。

要求：同一个学生选修的课程不能安排在同一时间进行内考试。

目标：用最短的时间组织完全部的考试。

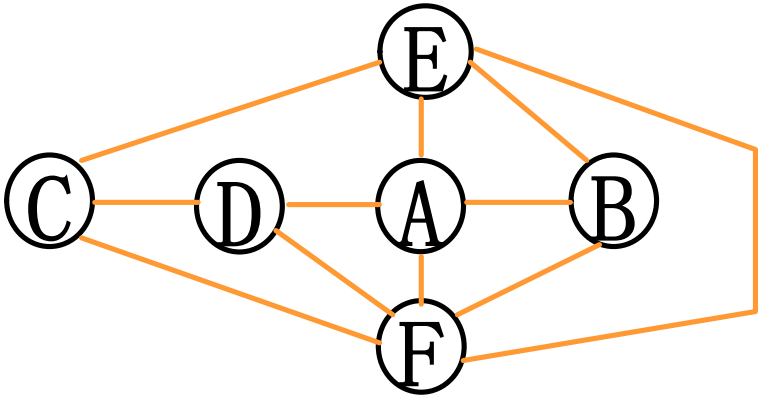
# 什么是数据结构

| 姓名 | 选修课1 | 选修课2 | 选修课3 |
| --- | --- | --- | --- |
| 刘德驰 | 算法分析（A) | 形式语言（B) | 计算机网络（E) |
| 周星华 | 计算机图形学（C) | 模式识别 （D) | |
| 李连龙 | 计算机图形学（C) | 计算机网络（E) | 人工智能（F) |
| 成　杰 | 模式识别（D ) | 人工智能（F) | 算法分析（A) |
| 令狐盈 | 形式语言（B) | 人工智能（F) | |



# 什么是数据结构

| 姓名 | 选修课1 | 选修课2 | 选修课3 |
| --- | --- | --- | --- |
| 刘德驰 | 算法分析（A） | 形式语言（B） | 计算机网络（E） |
| 周星华 | 计算机图形学（C） | 模式识别 （D） | |
| 李连龙 | 计算机图形学（C） | 计算机网络（E） | 人工智能（F） |
| 成　杰 | 模式识别（D ） | 人工智能（F） | 算法分析（A） |
| 令狐盈 | 形式语言（B） | 人工智能（F） | |

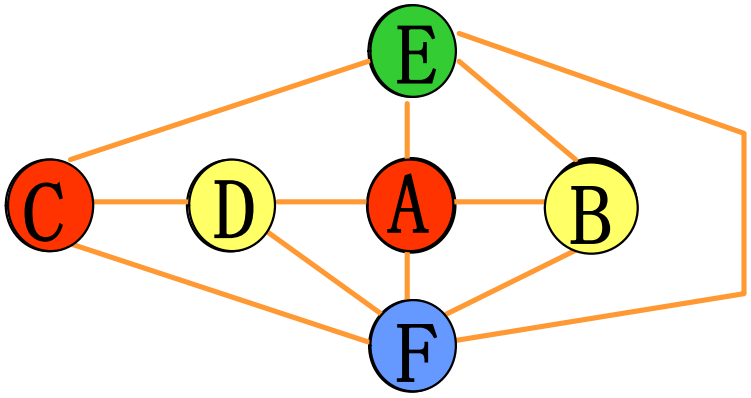


# 什么是数据结构

课程考试可考虑用图的着色法求解问题。
每一种颜色代表一个考试时间，相同颜色的顶点是可以安排在同一时间考试的课程；
用尽可能少的颜色为图的顶点着色，相邻的顶点着上不同的颜色。

**如下是一种可行的考试日程：**

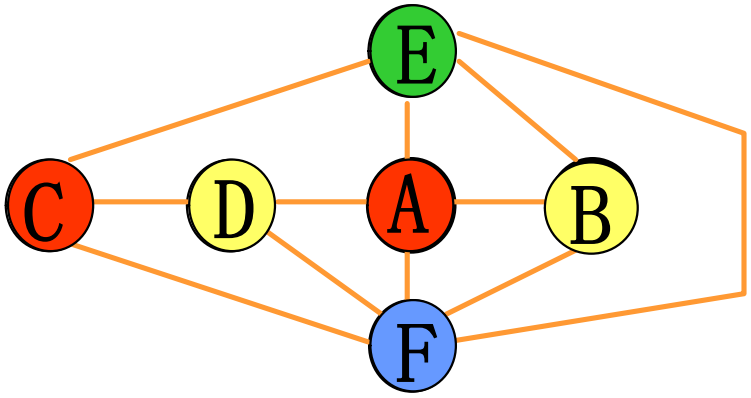
**1: 算法分析(A)**
**计算机图形学(C)**
**2: 形式语言(B)**
**模式识别人工智能（D)**
**3: 计算机网络(E)**
**4: 人工智能(F)**



# 什么是数据结构

设：G表示课程关系图，V是图G中所有尚未着色的顶点集合，NEW表示可以用新颜色着色的顶点集合。

**Step1：i=1；V ={ 图中所有顶点的集合 }**
**Step2：若 V 为空 则跳转到Step3，否则：**
**置 NEW 为空集合；**
**在 V 中取一点，找出所有与之“不相邻”的顶点；**
**将这些顶点加入 NEW，从 V 中去掉这些顶点**
**（第 i 天考试课程为 NEW 中顶点所对应的课程）**
**以输出NEW中顶点所对应的课程；**
**i ++；**
**跳转到Step2；**
**Step3：结束**



# 什么是数据结构

设：G表示课程关系图，V是图G中所有尚未着色的顶点集合，NEW表示可以用新颜色着色的顶点集合。

**Step1：i=1；V ={ 图中所有顶点的集合 }**
**Step2：若 V 为空 则跳转到Step3，否则：**
**置 NEW 为空集合；**
**在 V 中取一点，找出所有与之“不相邻”的顶点；**
**将这些顶点加入 NEW，从 V 中去掉这些顶点**
**（第 i 天考试课程为 NEW 中顶点所对应的课程）**
**以输出NEW中顶点所对应的课程；**
**i ++；**
**跳转到Step2；**
**Step3：结束**

集合是一种数据结构

# 什么是数据结构

数值求解问题

对象：len,wide,area ——用数值表示

对象之间的关系：

area = len × wide，——用方程或函数表示

数据存储：可用整型或实型变量存储数据

问题求解方法：某种数值计算方法求解

# 什么是数据结构

非数值问题

对象：课程——用符号/编码表示

对象之间的关系：存在某种关联的关系

数据存储：要保存数据及数据之间的关系

问题求解方法：很多很多

# 什么是数据结构

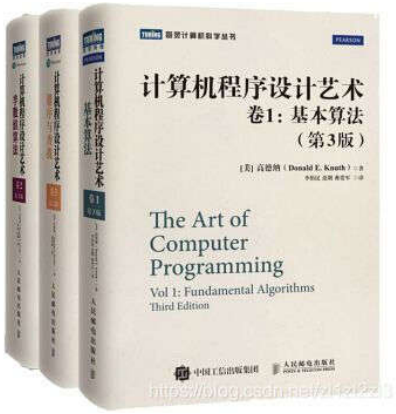
数值求解问题

《数值分析》

《数据结构》

非数值问题

# 什么是数据结构




1968年美国Donald.E.Knuth（唐纳德.E.克努特）教授出版：*The Art of Computer Programming*《计算机程序设计艺术》开创了数据结构的最初体系。计划共写7卷，然而出版三卷之后，已震惊世界，获得计算机科学界的最高荣誉图灵奖，年仅36岁。中文名：高德纳

# 什么是数据结构

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

什么是数据结构?

数据结构是一门研究非数值问题中计算机的操作对象以及它们之间的关系和操作的学科。

数据结构所研究的问题是什么?

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。

# 基本概念和术语

数据结构随着程序设计的发展而发展

- 无结构阶段
- 结构化阶段：数据结构 + 算法 = 程序
- 面向对象阶段：(数据结构 + 算法) = 程序

数据结构的发展并未终结

- 研究的范围不断扩展，算法不断更新
- 描述手段、使用语言不断更新

# 基本概念和术语

数据(data)：
客观对象的符号化表示。
数据元素(data element)：
数据的基本单位，在计算机程序中作为一个整体考虑和处理，通常具有完整确定的实际意义。(节点、顶点、记录)
数据项(data item)：
数据不可分割的最小标识单位。一个数据元素可由若干数据项组成，通常不具有完整确定的实际意义。(字段)

| | | | | |
|---|---|---|---|---|
| 0001 | 杨过 | 男 | 古墓派 | |
| 0002 | 令狐冲 | 男 | 华山派 | 恒山派兼职掌门 |
| 0003 | 程灵素 | 女 | 药王派 | |
| … | … | … | … | … |

# 基本概念和术语

数据结构(data structure)：相互之间存在一种或多种特定关系的、具有相同特征的数据元素的集合。

数据结构：带有结构和操作的数据元素集合。
结构：数据元素之间的关系；
操作：对数据的加工处理 。

# 基本概念和术语

数据结构的两个含义：

数据的逻辑结构：从具体问题抽象出来的数据模型，反映了事物的组成及事物之间的逻辑关系。

数据的存储结构：解决各种逻辑结构在计算机中的物理存储和表示，也称为物理结构。

同一种逻辑结构可以采用不同的表示方式，即采用不同的映射关系来建立数据的逻辑结构到存储结构的转换。

# 基本概念和术语

数据的逻辑结构：数据之间的结构关系，是具体关系的抽象。

逻辑结构一般有四种：

集合：数据元素间除“同属于一个集合”外，无其它关系

线性结构：一个对一个，如线性表/栈/队列

树形结构：一个对多个，如：树

图状结构：多个对多个，如：图

# 基本概念和术语

## 集合

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。按学生的学号顺序排列。

| 学号 | 姓名 | 专业 | 政治面貌 |
| --- | --- | --- | --- |
| 001 | 郭靖 | 计算机 | 党员 |
| 002 | 平一指 | 计算机 | 团员 |
| 003 | 乔峰 | 计算机 | 团员 |
| 004 | 胡一刀 | 计算机 | 团员 |
| 005 | 胡铁花 | 计算机 | 党员 |
| 006 | 沈浪 | 计算机 | 团员 |
| 007 | 张翠山 | 计算机 | 团员 |
| 008 | 陈近南 | 计算机 | 团员 |

# 基本概念和术语

## 线性结构

除第一个元素和最后一个元素外，其他元素都有且仅有一个直接前趋，有且仅有一个直接后继。

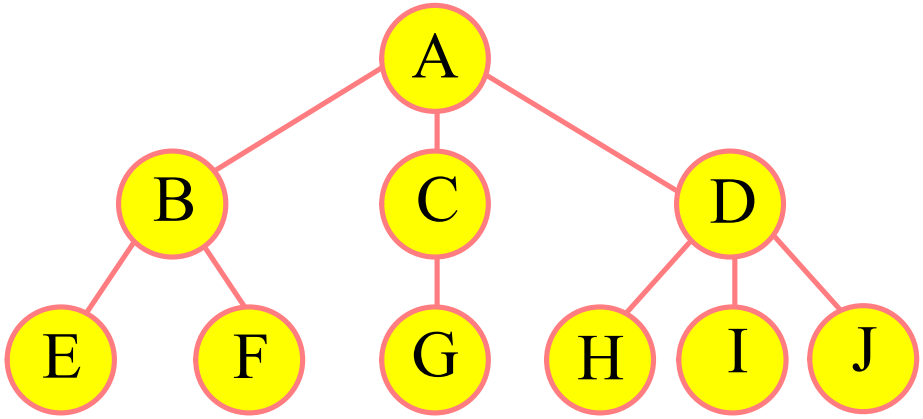
**视频里的每一帧**
**音乐里的每一个小节**
**食堂排队的队列**
**数组里的每一个元素**

| 学号 | 姓名 | 专业 | 政治面貌 |
|---|---|---|---|
| 001 | 宋远桥 | 计算机 | 党员 |
| 002 | 俞莲舟 | 计算机 | 团员 |
| 003 | 俞岱岩 | 计算机 | 团员 |
| 004 | 张松溪 | 计算机 | 团员 |
| 005 | 张翠山 | 计算机 | 党员 |
| 006 | 殷梨亭 | 计算机 | 团员 |
| 007 | 莫声谷 | 计算机 | 团员 |

# 基本概念和术语

## 树形结构

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图表示。

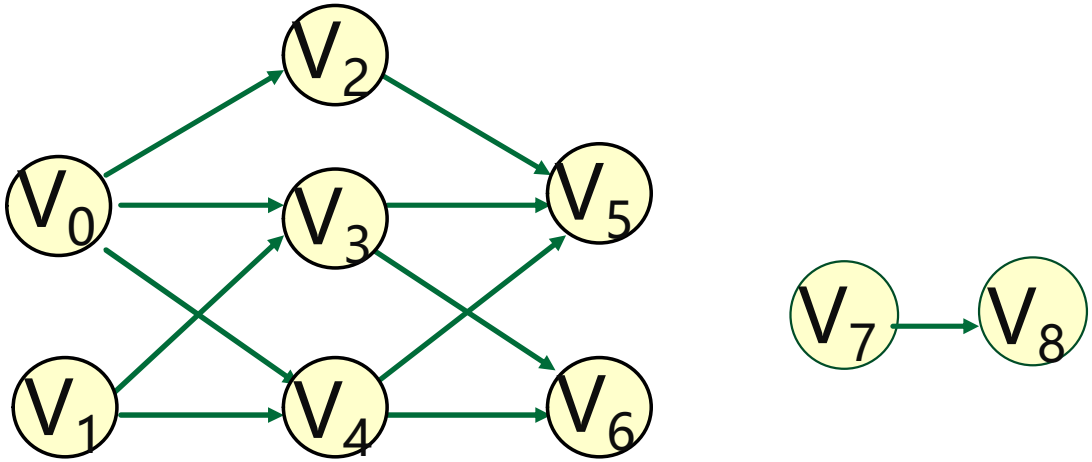



**每一个元素只有一个直接前趋，有0个或多个直接后继。**

# 基本概念和术语

图形结构

人际关系图。




**每一个元素可以有0个或多个直接前趋，有0个或多个直接后继。**

## 数据结构的表示方法

二元组表示

二元组表示是用一个二元组（D，S）表示数据结构，其中 D 是数据元素集合，S 是 D 上关系的集合。

# 基本概念和术语

集合关系

例：学生基本情况表的二元组表示（D，S）

| 学号 | 姓名 | 专业 | 政治面貌 |
| --- | --- | --- | --- |
| 001 | 郭靖 | 计算机 | 党员 |
| 002 | 平一指 | 计算机 | 团员 |
| 003 | 乔峰 | 计算机 | 团员 |
| 004 | 胡一刀 | 计算机 | 团员 |
| 005 | 胡铁花 | 计算机 | 党员 |
| 006 | 沈浪 | 计算机 | 团员 |
| 007 | 朱七七 | 计算机 | 团员 |
| 008 | 李莫愁 | 计算机 | 团员 |

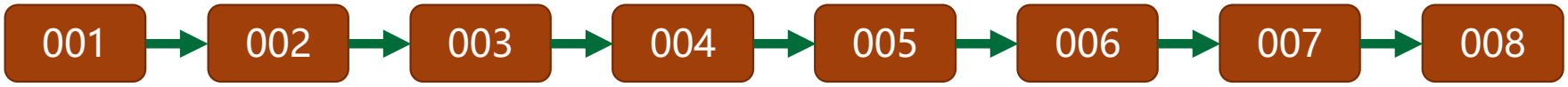
**D = { 001，002，003，004，005，006，007，008 }**

**S = { R }**

**R = {}**

# 基本概念和术语

例：学生基本情况表的二元组表示（D，S）



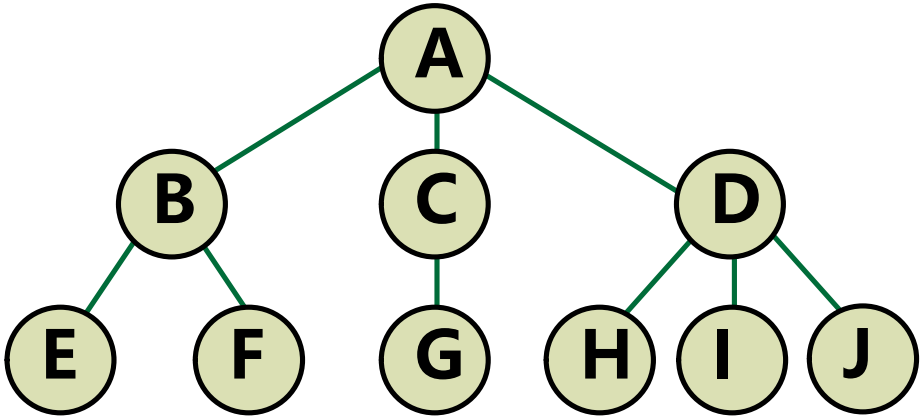

D = { 001, 002, 003, 004, 005, 006, 007, 008 }

S = { R }

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

## 基本概念和术语

例：家族树的二元组表示（D，S）



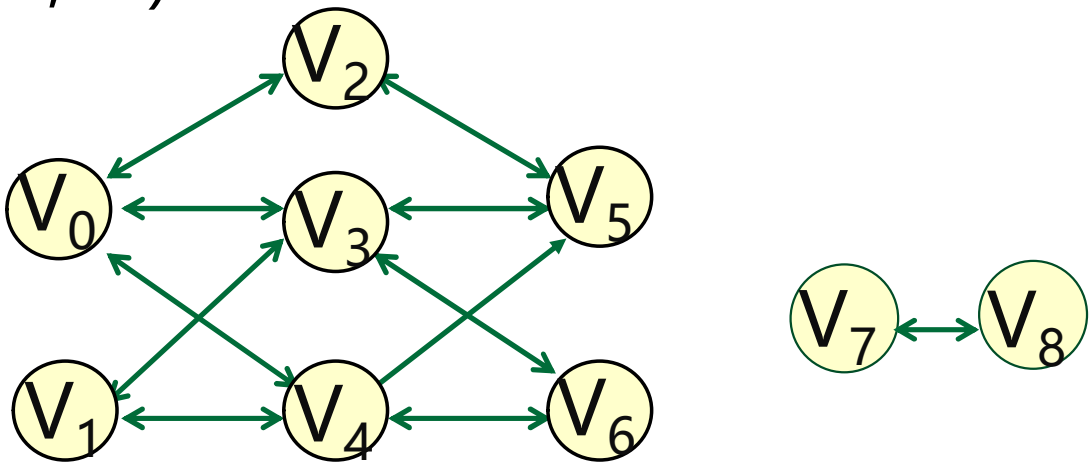

D = { A，B，C，D，E，F，G，H，I，J }

S = { R }

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

# 基本概念和术语

例：人际关系图的二元组表示（D，S）




$D = \{V_0, V_1, V_2, V_3, V_4, V_5, V_6, V_7, V_8\}$

$S = \{R\}$

$R = \{ \langle V_0,V_2\rangle, \langle V_2,V_0\rangle, \ldots\ldots, \langle V_3,V_5\rangle, \langle V_7,V_8\rangle, \langle V_8,V_7\}$

## 基本概念和术语

逻辑结构：从操作对象抽象出来的数学模型。

存储结构：逻辑结构在计算机中的表示,存储结构的实质是内存分配，在具体实现时，依赖于计算机语言。

1. 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；
2. 数据的存储结构属于具体实现的视图，是面向计算机的。
3. 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。

# 基本概念和术语

- 常见的存储结构
  - 顺序存储方式
  - 链式存储方式
  - 散列方式
  - 索引方式。

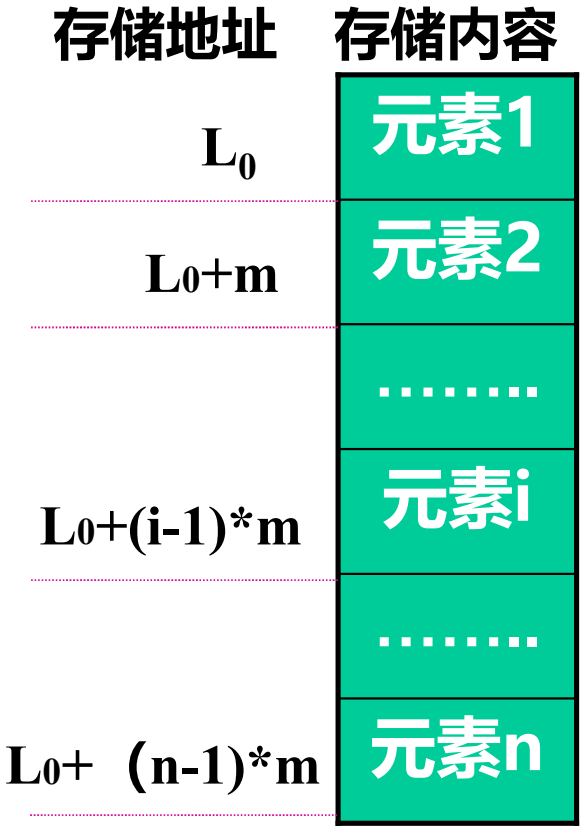
## 基本概念和术语

顺序存储方式：借助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。例如，用一维数组存储线性结构。

链式存储方式：借助指示元素存储地址的指针来表示数据元素之间的逻辑关系。例如，用链表(指针)存储线性结构。

# 基本概念和术语

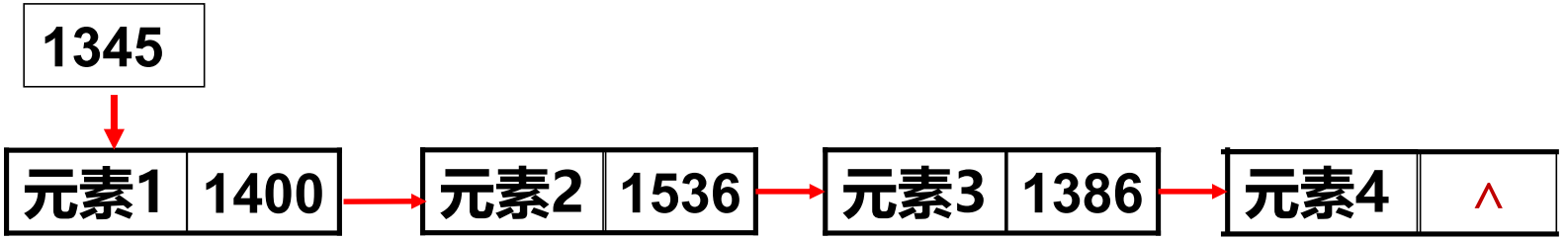
顺序存储结构




Location(**元素**i) = $L_0$ + (i-1) * m

# 基本概念和术语

链式存储结构




| 存储地址 | 存储内容 | 指针 |
|---|---|---|
| 1345 | 元素1 | 1400 |
| 1386 | 元素4 | ^ |
| ……. | …….. | ……. |
| 1400 | 元素2 | 1536 |
| ……. | …….. | ……. |
| 1536 | 元素3 | 1386 |

# 抽象数据类型

模块化思想的发展，为模块的划分提供了理论依据，简称ADT (Abstract Data Type)

可以看作是定义了一组操作的一个抽象模型

例如，集合与集合的并、交、差运算就可定义为一个的抽象数据类型

一个抽象数据类型要包括哪些操作，这一点由设计者根据需要确定

例如，对于集合，如果需要，也可以把判别一个集合是否为空集或两个集合是否相等作为集合上的操作

# 抽象数据类型

ADT是一种描述用户与数据之间接口的抽象模型。它定义了一个数据模型和在这个数据模型上的一组操作。

**抽象数据类型的定义仅取决于它的一组逻辑特性，而与其在计算机内部如何表示和实现无关，即不论其内部结构如何变化，只要它的数学特性不变，都不影响其外部的使用。**

**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>**
**}**

# 抽象数据类型

ADT List {

数据对象：D = { $a_1$, $a_2$, … , $a_{i-1}$, $a_i$, $a_{i+1}$, …, $a_n$ }

数据关系：R = { <$a_1$,$a_2$>,<$a_2$,$a_3$>,…,<$a_{n-1}$,$a_n$> }

基本操作： <基本操作的定义>

InitList(&L)

操作结果： 构造一个空的线性表L；

DetroyList(&L)

初始条件：线性表已存在

操作结果：销毁线性表L

ClearList(&L)

初始条件：线性表已存在

操作结果：将L重置为空表

ListEmpty(L)

初始条件：线性表已存在

操作结果：若L为空表返回TRUE，否则返回FALSE

…

}

# 算法与算法分析

算法是为了解决某类问题而规定的一个有限长的操作序列。

一个算法必须满足以下五个重要特性：有穷性、确定性、可行性、有输入和有输出。

- 有穷性：算法必须在有限步内结束。
- 确定性：算法的操作清晰无二义性。
- 可行性：算法的操作必须能够实现。
- 有输入：有0个或多个输入。
- 有输出：有1个或多个输出。

# 算法与算法分析

算法设计与算法分析是计算机科学的核心问题。

常用的设计方法

- 穷举法　(百钱买百鸡)
- 贪心法　(Huffman树)
- 递归法, 分治法 (二分法检索)
- 回溯法 (八皇后)
- 动态规划法 (最佳二叉排序树)
- 宽度优先和深度优先搜索
- 分枝界限法
- ……

# 算法与算法分析

评价算法的标准

正确性，可读性，可维护性，健壮性，效率。

算法效率的度量

程序所用算法运行时所要花费的时间代价

程序中使用的数据结构占有的空间代价

算法的时间复杂度：算法的时间效率

算法的空间复杂度：算法的空间效率

# 算法与算法分析

评价算法效率的方法

事后统计法

通过上机运行，测试算法花费的时间。算法的平均执行时间，算法的最大执行时间。

缺点：

1) 必须编写、执行程序

2）其它因素掩盖算法本质（比如机器比较慢）

事前分析估算法

# 算法与算法分析

影响算法时间效率的因素

- 算法选用的策略
- 问题的规模
- 编写程序的语言
- 编译程序产生的机器代码的质量
- 计算机运行速度

事前分析估算法

对算法所需要的计算机资源——时间和空间进行估算。

# 算法与算法分析



算法分析

感兴趣的不是具体的资源占用量，而是与具体的平台无关、具体的输入实例无关，且随输入规模增长的值是可预测的。

与问题的规模之间的关系，用一定“规模(size)”的数据作为输入时程序运行所需的“基本操作(basic operation)” 数来描述时间效率。

# 算法与算法分析

算法的渐进分析就是要估计，当数据规模n逐步增大时，资源开销 f(n) 的增长趋势

从数量级大小的比较来考虑，当 n 增大到一定值以后，资源开销的计算公式中影响最大的就是 n 的幂次最高的项，其他的常数项和低幂次项都是可以忽略的

# 算法与算法分析

北京理工大学 BEIJING INSTITUTE OF TECHNOLOGY

## 算法分析

由于算法的复杂性与其所求解的问题规模直接有关，因此通常将问题规模 n 作为一个参照量，求算法的时空开销与 n 的关系

一般这种函数关系都相当复杂，计算时只考虑可以显著影响函数量级的部分，即结果为原函数的一个近似值

对资源开销的一种不精确估计，提供对于算法资源开销进行评估的简单化模型

**渐进分析法：**

$$f(\mathrm{n}) = n^2 + 100\mathrm{n} + \log_{10} n + 1000$$

**适用于上万个数据规模的算法未必适用于只有10个数据的算法**

## 大*O*表示法

定义1：如果存在整数c和N，使得对任意的n>=N,都有f(n)<=cg(n),则称f(n)在集合*O(g(n))中，或简称f(n)是O(g(n)) 的*

该定义说明了函数f和g之间的关系，既可以说成函数g(n)是函数f(n)取值的上界，也可以说是函数f的增长最多趋同于函数g的增长。

大$O$表示法

定义2：存在一个整数c和N，使得对任意的n>=N,满足

$$\lim_{N\to\infty}\left|\left(\frac{T(n)}{f(n)}\right)\right| = c$$

这表明随着数据规模n逐渐增大，T(n)算法的执行时间的增长率和f(n)的增长率相同。

举个例子：

输入n个数，求解1到n的和。

举个例子：

查找n个数的集合中是否存在m。

举个例子：

查找n个数的有序集合中是否存在m。

举个例子：

对n个数字进行排序（冒泡排序）。

举个例子：

对n个数字进行排序（快排排序）。

举个例子：

斐波拉契数列1。

举个例子：

斐波拉契数列2。

举个例子：

斐波拉契数列3-采用特征方程求解。

# 算法与算法分析

$a_1 = a_2 = 1$ ， $a_{n+2} = a_{n+1} + a_n \ (n \in N_+)$

整理得： $a_{n+2} - a_{n+1} - a_n = 0$

其特征方程为： $x^2 - x - 1 = 0$

$$f(n) = c_1 \left(\frac{1+\sqrt{5}}{2}\right)^n + c_2 \left(\frac{1-\sqrt{5}}{2}\right)^n$$

解得： $\begin{cases} x_1 = \frac{1+\sqrt{5}}{2} \\ x_2 = \frac{1-\sqrt{5}}{2} \end{cases}$

$$f(n) = \frac{1}{\sqrt{5}} \left[ \left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n \right]$$

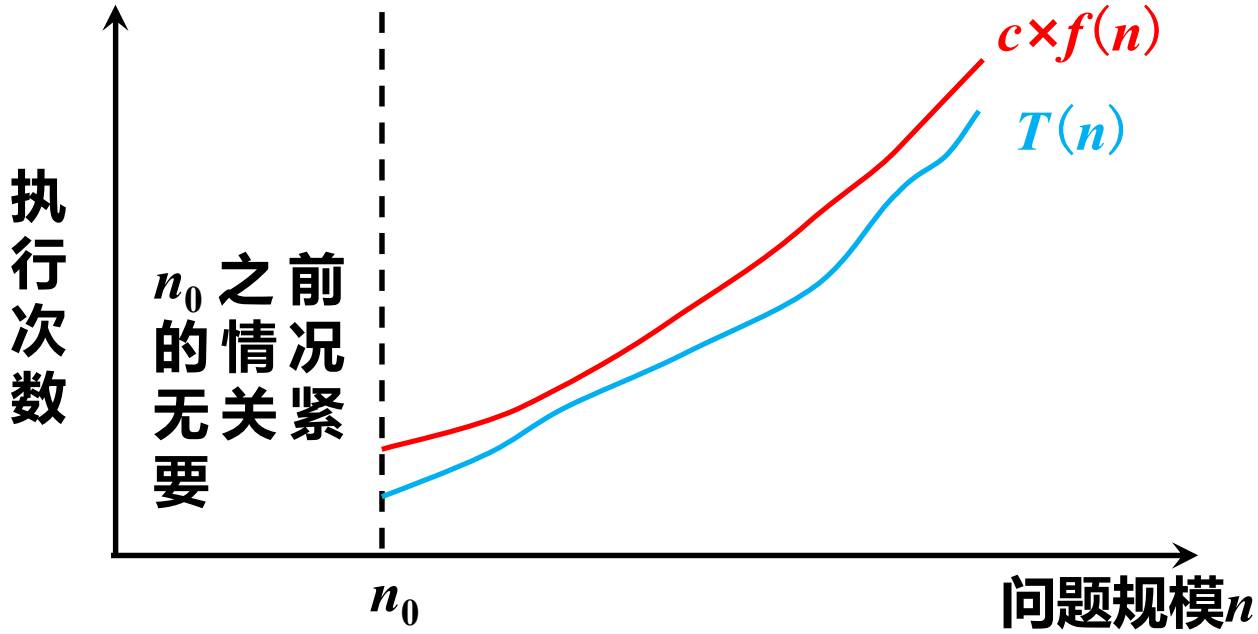
```
int fib3(int n) {
  double c = sqrt(5);
  return  (int) ( ( pow((1+c)/2,n) - pow((1-c)/2,n) ) /c ) ;
}
```

## 大$O$表示法

例如一个在集合O(n)中的函数，也一定在集合$O(n^2)$中，同时也在集合$O(n^3)$中。

大$O$表示法给出了在所有上限中最小的那个上限。

# 大O表示法



# 算法与算法分析

算法由控制语句和原操作（预定义数据类型的操作）组成

原操作是算法的核心操作

例：n 阶矩阵相乘算法

```
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 ];
    }
```

## 算法与算法分析

例：n 阶矩阵相乘算法

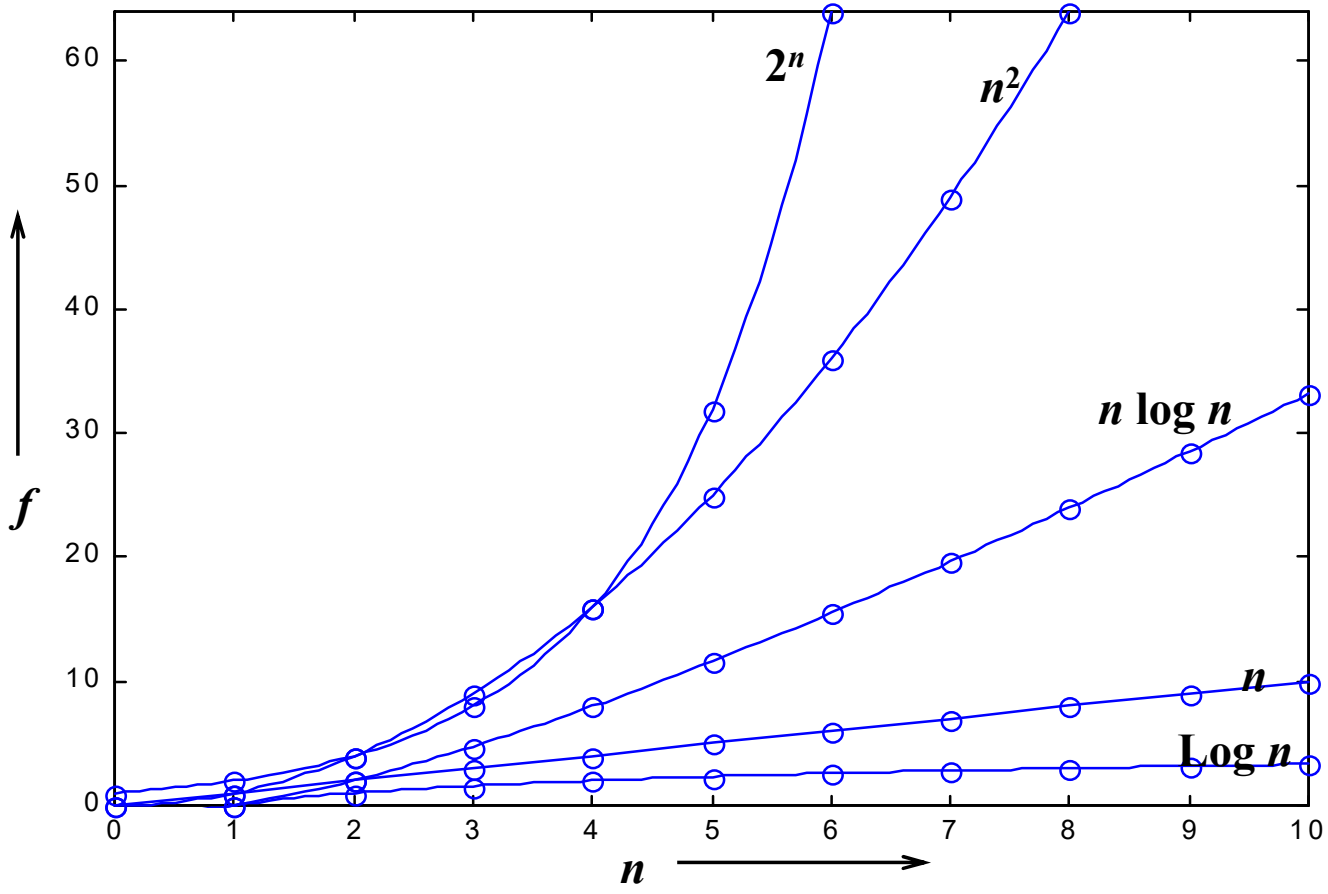
```
for ( i = 1; i<=n; ++i )
    for (j = 1; j<=n; ++j ) {
        c[ i ][ j ] = 0 ;
        for (k = 1; k<= n; ++k )
          c[ i ][ j ] += a[ i ][ k ] * b[ k] [ j ]
    }
```

$T(n)= n^2 +n^3$

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

**在计算算法时间复杂度时，可以忽略所有低次幂和最高次幂的系数。**

# 算法与算法分析



# 算法复杂性的不同数量级的变化

| n | $lon_2n$ | $nlog_2n$ | $n^2$ | $n^3$ | $2^n$ | n! |
|---|---|---|---|---|---|---|
| 4 | 2 | 8 | 16 | 64 | 16 | 24 |
| 8 | 3 | 24 | 64 | 512 | 256 | 80320 |
| 10 | 3.32 | 33.2 | 100 | 1000 | 1024 | 3628800 |
| 16 | 4 | 64 | 256 | 4096 | 65536 | $2.1*10^{13}$ |
| 32 | 5 | 160 | 1024 | 32768 | $4.3*10^9$ | $2.6*10^{35}$ |
| 128 | 7 | 896 | 16384 | 2097152 | $3.4*10^{38}$ | ∞ |
| 1024 | 10 | 10240 | 1048576 | $1.07*10^9$ | ∞ | ∞ |
| 10000 | 13.29 | 132877 | $10^8$ | $10^{12}$ | ∞ | ∞ |

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

# 算法与算法分析

- 大O表示法的运算规则
  - 单位时间
    - 简单布尔或算术运算
    - 简单I/O
    - 函数返回
  - 加法规则：$f_1(n)+f_2(n)=O(max(f_1(n), f_2(n)))$
    - 顺序结构，if 结构，switch结构
  - 乘法规则：$f_1(n)\cdot f_2(n) = O(f_1(n)\cdot f_2(n))$
    - for, while, do-while结构

# 算法与算法分析

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n个元素且每一个元素值都为 k，或不为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与问题规模有关?

## 最好情况、最好情况、平均情况

如果问题规模相同，时间代价与输入数据有关，则需要分析最好情况、最坏情况和平均情况。

**最好情况：出现概率较大时分析**

**最差情况：出现概率较大时分析**

**平均情况：已知输入数据是如何分布的，通常假设等概率分布**

# 算法与算法分析

算法空间

固定空间

输入数据、结果数据、程序代码所需的空间

可变空间(辅助空间)

中间结果、递归调用所需的空间

算法所需的辅助空间度量

假如随着问题规模 n 的增长，算法运行所需辅助空间 S(n) 的增长率与 g(n) 的增长率相同，则记作：S(n)=O(g(n))，称O(g(n)) 为算法的空间复杂度。

# 算法与算法分析

```
void bubble_sort (int a[ ], int n )
 {  //冒泡排序
    for (i = n-1,change=TRUE; i >0&&change; - -i )
    {   change= FALSE;
        for (j = 0; j<= i; ++j )
            if (a [j] >a[ j+1])
            { temp=a [j]; a[j]= a[ j+ 1]; a[j+1]=temp;
              change=TRUE;
            }
    }
}
```

$S(n) = O(1)$

例题1. 算法的时间复杂度与（　）有关。

A. 问题规模　　B. 计算机硬件的运行速度

C. 源程序的长度　　D. 编译后执行程序的质量

例题1. 算法的时间复杂度与（　）有关。

A. 问题规模　　B. 计算机硬件的运行速度

C. 源程序的长度　　D. 编译后执行程序的质量

解答：A。算法的具体执行时间与计算机硬件的运行速度、编译产生的目标程序的质量有关，但这属于事后测量。算法的时间复杂度的度量属于事前估计，与问题的规模有关。

例题2. 某算法的时间复杂度是$O(n^2)$，表明该算法（　　）。

A. 问题规模是$n^2$　　B. 问题规模与$n^2$成正比

C. 执行时间等于$n^2$　　D. 执行时间与$n^2$成正比

例题2. 某算法的时间复杂度是$O(n^2)$，表明该算法（　）。

A. 问题规模是$n^2$　　B. 问题规模与$n^2$成正比

C. 执行时间等于$n^2$　　D. 执行时间与$n^2$成正比

解答：D。算法的的时间复杂度是$O(n^2)$，这是设定问题规模为 n 的分析结果，所以A、B 都不对；它也不表明执行时间等于$n^2$，它只表明算法的执行时间$T(n)\leq c\times n^2$（c为比例常数）。有的算法，如$n\times n$矩阵的转置，时间复杂度为$O(n^2)$，不表明问题规模是$n^2$。

例题3. 下面说法中错误的是（　）。

① 算法原地工作的含义是指不需要任何额外的辅助空间

② 在相同问题规模 n 下，时间复杂度为$O(n)$的算法总是优于时间复杂度为$O(2^n)$的算法

③ 所谓时间复杂度是指在最坏情形下估算算法执行时间的一个上界

④ 同一个算法，实现语言的级别越高，执行效率越低

A. ①　　B. ①②　C. ①④　D. ③

例题3. 下面说法中错误的是（　）。

① 算法原地工作的含义是指不需要任何额外的辅助空间

② 在相同问题规模 n 下，时间复杂度为$O(n)$的算法总是优于时间复杂度为$O(2^n)$的算法

③ 所谓时间复杂度是指在最坏情形下估算算法执行时间的一个上界

④ 同一个算法，实现语言的级别越高，执行效率越低

A. ①　　B. ①②　C. ①④　D. ③

解答：A。算法原地工作的含义指空间复杂度$O(1)$

例题4. 有实现同一功能的两个算法$A_1$和 $A_2$，其中$A_1$ 的渐进时间复杂度是$T_1(n) = O(2^n)$，$A_2$ 的渐进时间复杂度是$T_2(n) = O(n^2)$。仅就时间复杂度而言，具体分析这两个算法哪个好。

例题4. 有实现同一功能的两个算法$A_1$和 $A_2$，其中$A_1$ 的渐进时间复杂度是$T_1(n) = O(2^n)$，$A_2$ 的渐进时间复杂度是$T_2(n) = O(n^2)$。仅就时间复杂度而言，具体分析这两个算法哪个好。

解答：比较算法好坏需比较两个函数$2^n$和$n^2$。

当$n = 1$时，$2^1 > 1^2$，算法$A_2$好于$A_1$

当$n = 2$时，$2^2 = 2^2$，算法$A_1$与$A_2$相当

当$n = 3$时，$2^3 < 3^2$，算法$A_1$好于$A_2$

当$n = 4$时，$2^4 = 4^2$，算法$A_1$与$A_2$相当

当$n > 4$时，$2^n > n^2$，算法$A_2$好于$A_1$

当$n \to \infty$时，算法$A_2$在时间上优于$A_1$。

例题5　设n是描述问题规模的非负整数，下面程序片段的时间复杂度是

```
x = 2;
while ( x < n/2 )
    x = 2*x;
```

A. $O(\log_2 n)$　B. $O(n)$　C. $O(n\log_2 n)$　D. $O(n^2)$

例题5　设n是描述问题规模的非负整数，下面程序片段的时间复杂度是

```
x = 2;
while ( x < n/2 )
    x = 2*x;
```

A. $O(\log_2 n)$　B. $O(n)$　C. $O(n\log_2 n)$　D. $O(n^2)$

解答：选A。找关键操作，即最内层循环中的执行语句 x = 2*x。因为每次循环x都成倍增长，设 $x = 2^k < n/2$，$2^{k+1} < n$，则 $k < \log_2 n - 1$，实际while循环内的语句执行了$\log_2 n - 2$次。

例题6 求整数n（n≥0）阶乘的算法如下，其时间复杂度是

```
int fact ( int n ) {
    if ( n <= 1 ) return 1;
    return n*fact ( n-1 );
}
```

A. $O(\log_2 n)$　B. $O(n)$　C. $O(n\log_2 n)$　D. $O(n^2)$

例题6 求整数n（n≥0）阶乘的算法如下，其时间复杂度是

```
int fact ( int n ) {
        if ( n <= 1 ) return 1;
    return n*fact ( n-1 );
}
```

A. $O(\log_2 n)$　B. $O(n)$　C. $O(n\log_2 n)$　D. $O(n^2)$

## 算法与算法分析

例题6 求整数n（n≥0）阶乘的算法如下，其时间复杂度是

```
int fact ( int n ) {
    if ( n <= 1 ) return 1;
    return n*fact ( n-1 );
}
```

A. $O(\log_2 n)$　B. $O(n)$　C. $O(n\log_2 n)$　D. $O(n^2)$

解答：选B，因为T(1) = 1，T(n) = 1+T(n-1) = 1+(1+T(n-2))
= 2+(1+T(n-3)) = 3+(1+T(n-4)) =

... = (n-1)+T(1) = n

# 总结

**数据结构：带有结构和操作的数据元素集合**

- 结构
  - 数据的逻辑结构
    - 线性结构
      - 线性表
      - 栈
      - 队列
    - 非线性结构
      - 树形结构
      - 图形结构
  - 数据的存储结构
    - 顺序存储
    - 链式存储
    - 索引存储
    - 散列存储
- 操作
  - 数据操作的定义（功能）
  - 数据操作的实现（在计算机上）

**算法效率**