

第1章 绪论

主讲：徐小华



- Office: 西区电3楼522
- Email: xiaohuaxu@ustc.edu.cn
- 微信号: 13035065278

为什么要学习数据结构

- ❖ 编程基础
- ❖ 计算机及相关专业考研考博课程
- ❖ 计算机等级考试课程
- ❖ 程序员考试课程

课程学习指导

课程特点：内容抽象、概念性强、内容灵活、不易掌握

- 1.提前预习、认真听课、按时完成书面及上机作业
- 2.注意先修课程的知识储备
 - ✓ C语言（递归；跟踪调试）
- 3.注意循序渐进：
 - ✓ 基本概念、基本思想、基本步骤、算法设计
- 4.注意培养算法设计的能力
 - ✓ 理解所讲算法、多做思考：若问题要求不同，应如何选择数据结构，设计有效的算法

课程基本信息

- 教材:《数据结构》(c 语言版)(第二版), 严蔚敏等编著, 人邮出版社
- 参考书目
 - 《数据结构题集 (c 语言版) 》, 严蔚敏等, 清华大学出版社
 - 《数据结构》陈越、何钦铭等, 高等教育出版社
 - "Data Structures with C++" Williaw Ford et al., Prentice Hall Inc.
 - "Data Structures Program Design in C, 2nd Ed." Robert Kruse et al., Prentice Hall Inc.
- 学时/学分: 理论课时 60; 实验课时 40; 学分 4
- 上课时间: 1-16 周, 周二下午 (8, 9)、周四下午 (6, 7)
- 上课地点: 3B101
- 考核方式: 统一闭卷考试, 成绩计算: 作业 + 上机 + 期末考试

主讲：徐小华 xiaohuaxu@ustc.edu.cn

QQ号：2532471830

助教：

蔡孟辛 PB2211

有问题的同学可随时联系我和助教，email回复时间一般在当天或第二天，微信基本在线😊！

考核方式及约法三章

- 平时成绩：18%+28%+4%
 - 18%：平时作业
 - 28%：上机实验
 - 平时作业和上机独立完成。不交作业的话后果很严重！
 - 4%：考勤
 - 无故旷课^①：每次扣1分，扣满5分后果很严重！
- 期末成绩：50%(闭卷笔试)
 - 总评计算方法：……（这是咱们之间的小秘密😊）

①：个性化修读的同学请联系助教报备，以免考勤扣分

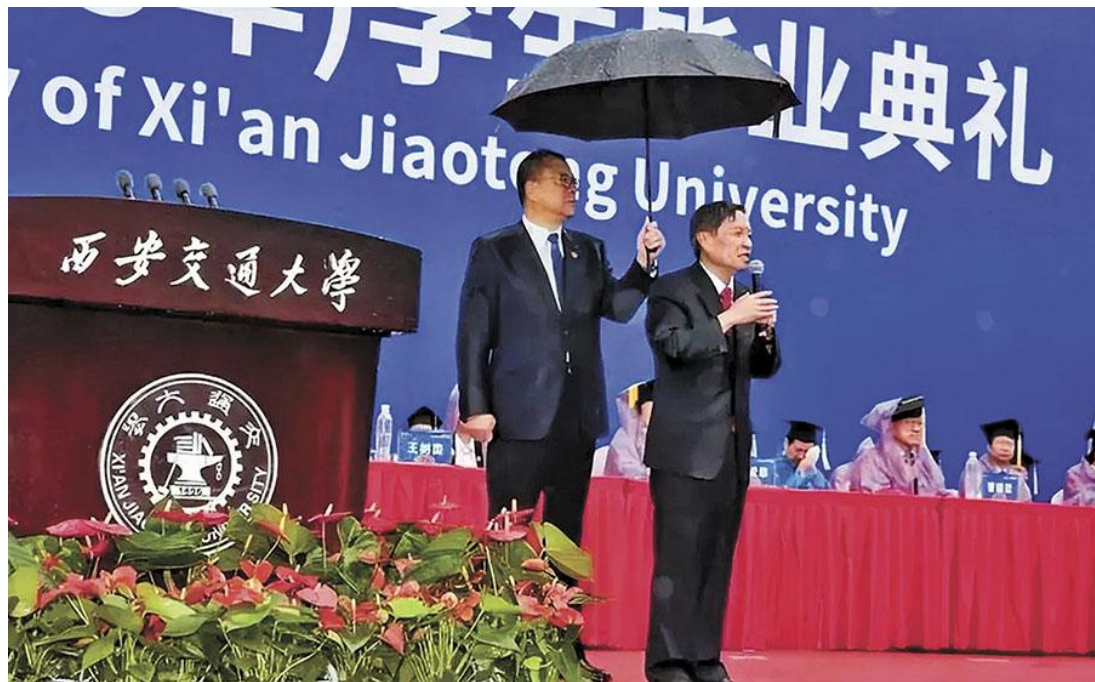


教学目标

1. 了解数据结构研究的主要内容
2. 掌握数据结构中涉及的基本概念
3. 掌握算法的时间、空间复杂度及其分析的简易方法

学生为校长撑伞





2023年7月2日上午，西安交通大学举行2023届研究生毕业典礼。在校长王树国为毕业生致辞时，突然大雨倾盆。西安交大党委书记卢建军为王树国撑起了雨伞

教学内容

- 1.1 数据结构的研究内容
- 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 算法与算法分析



1.1 数据结构的研究内容

📖 N.沃思 (Niklaus Wirth)教授提出:

程序=算法+数据结构

📖 电子计算机的主要用途:

☞ 早期:

主要用于数值计算

☞ 后来:

处理逐渐扩大到非数值计算领域, 能处理多种复杂的具有一定结构关系的数据

书目自动检索系统

线性表

书目文件

001	高等数学	樊映川	S01
002	理论力学	罗远祥	L01
003	高等数学	华罗庚	S01
004	线性代数	栾汝书	S02
.....	作者名:...

索引表

按书名

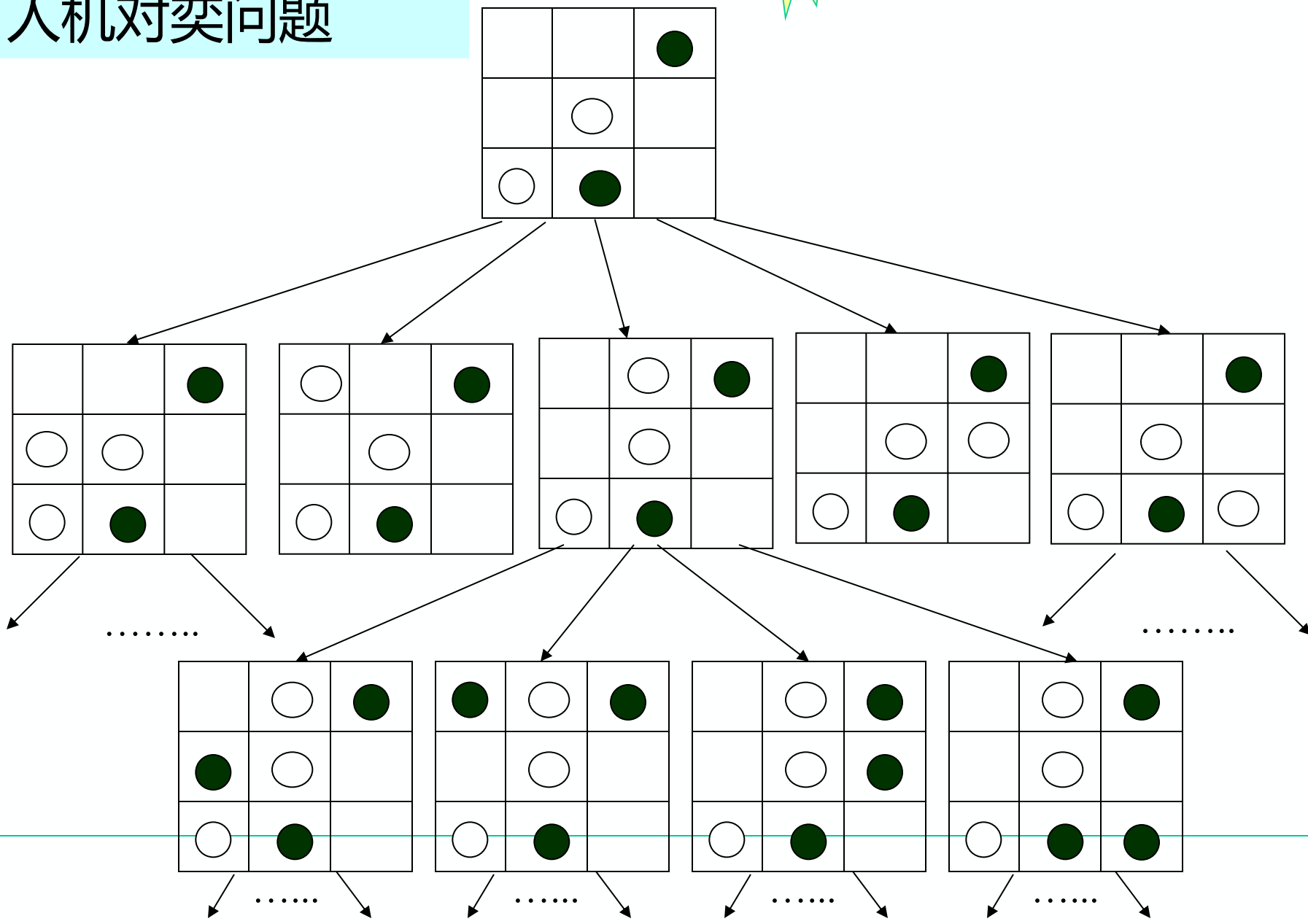
高等数学	001, 003	出版单位: 樊映川	001, ...
理论力学	002, ...	出版时间: 华罗庚	002, ...
线性代数	004, ...	栾汝书	004, ...
.....

按作者名

按分类号

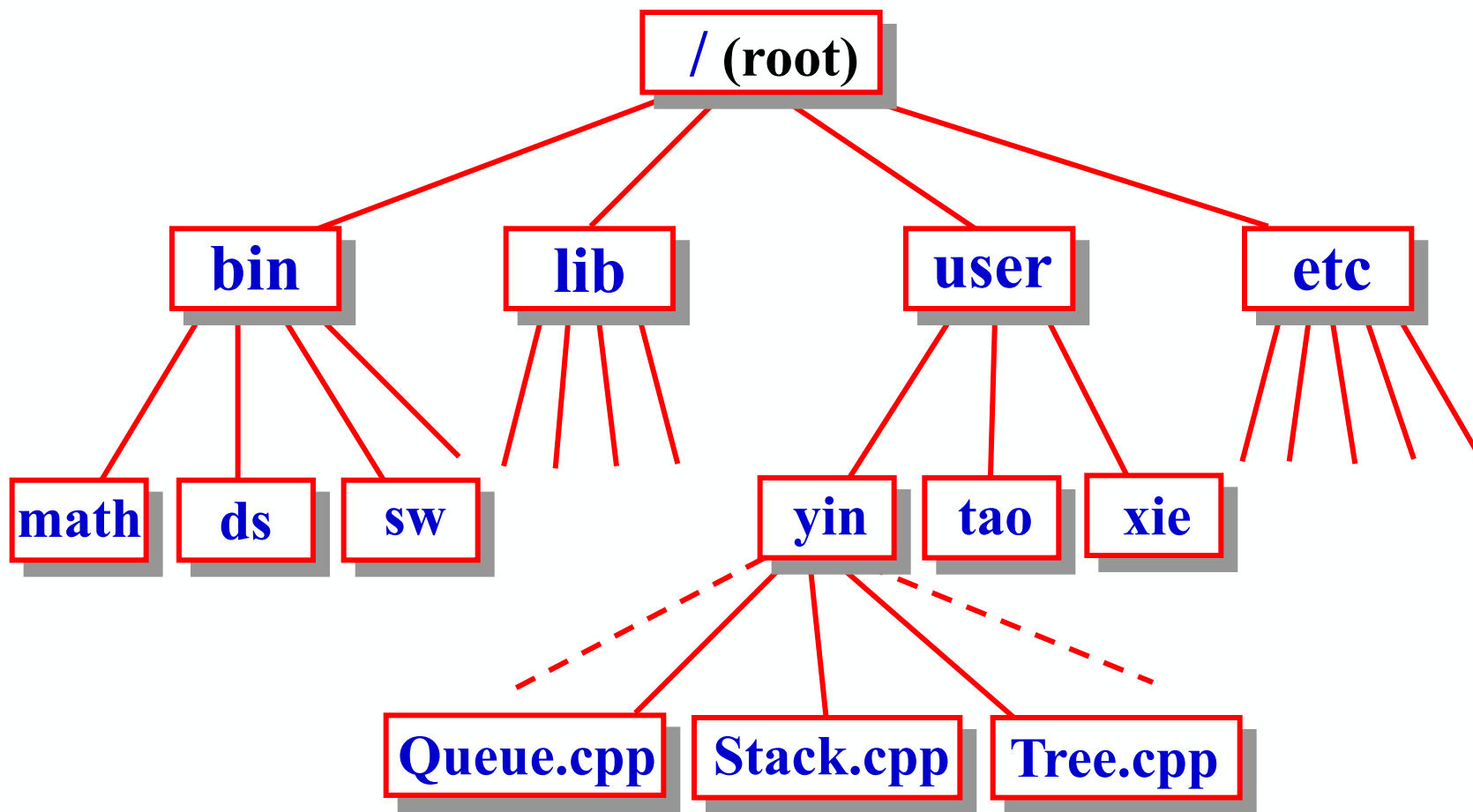
L	002, ...
S	001, 003,
.....

人机对奕问题



文件系统的系统结构图

树



田径赛的时间安排问题

图

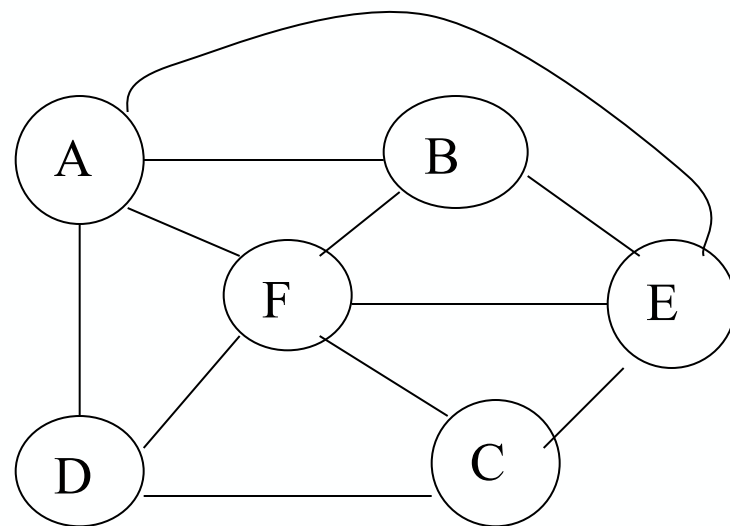
假设共有六个比赛项目，规定每个选手至多可参加三个项目，有五人报名参加比赛（如下表所示）设计比赛日程表，使得在尽可能短的时间内完成比赛。

姓 名	项目 1	项目 2	项目 3
丁 一	跳高	跳 远	100 米
马 二	标 枪	铅 球	
张 三	标 枪	100 米	200 米
李 四	铅 球	200 米	跳 高
王 五	跳 远	200 米	

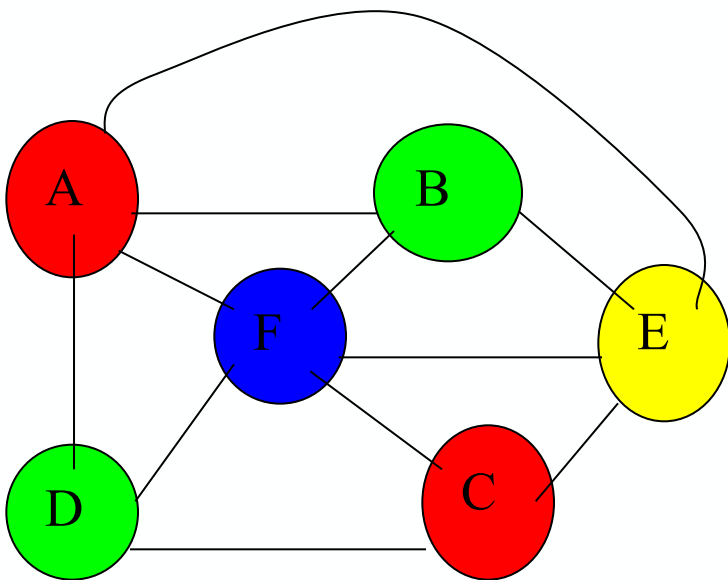
田径赛的时间安排问题

图

姓名	项目1	项目2	项目3
丁一	A	B	E
马二	C	D	
张三	C	E	F
李四	D	F	A
王五	B	F	



姓名	项目1	项目2	项目3
丁一	A	B	E
马二	C	D	
张三	C	E	F
李四	D	F	A
王五	B	F	

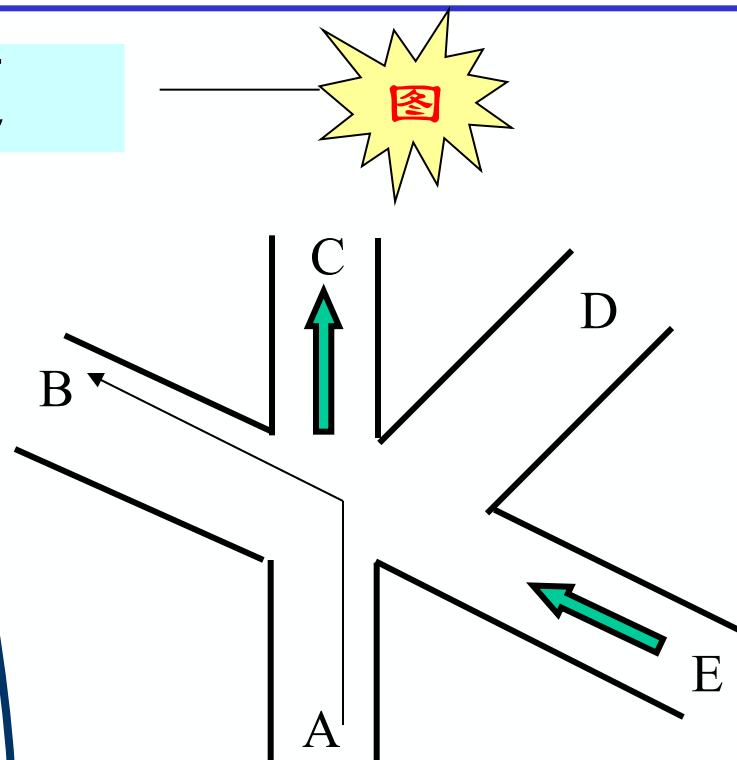
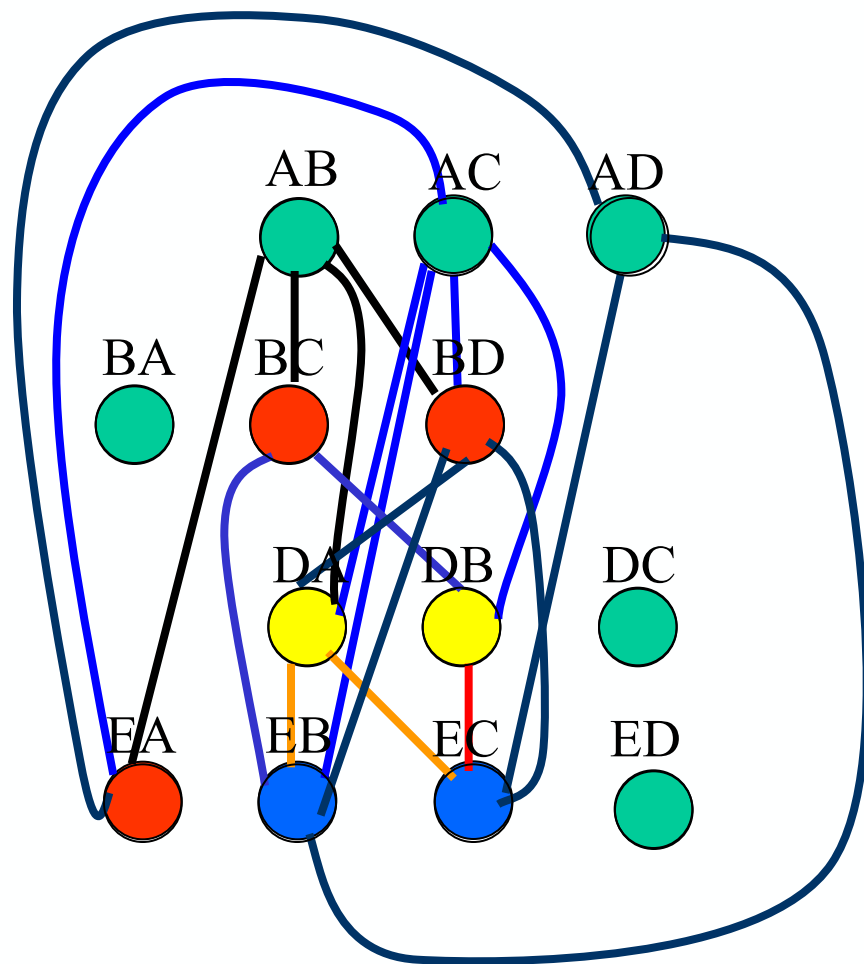


只需
安排四
个单位
时间进
行比赛

比赛时 间段	比赛项目
1 (红)	A, C
2 (绿)	B, D
3 (黄)	E
4 (蓝)	F

本质上是一个 “**图**着色问题”

多叉路口交通灯管理问题



顶点：一条通路
连线：不能同时通行
染色：有连线的两个顶点不能具有相同颜色

 求解非数值计算的问题：

设计出合适的数据结构及相应的算法

即：首先要考虑对相关的各种信息如何表示、组织和存储？

数据结构的研究内容为：

研究非数值计算的程序设计问题中计算机的**操作对象**以及它们之间的**关系和操作**。

数据结构课程的形成和发展:

形成阶段:

60年代初期, “数据结构”有关的内容散见于操作系统、编译原理和表处理语言等课程。1968年, “数据结构”被列入美国一些大学计算机科学系的教学计划。

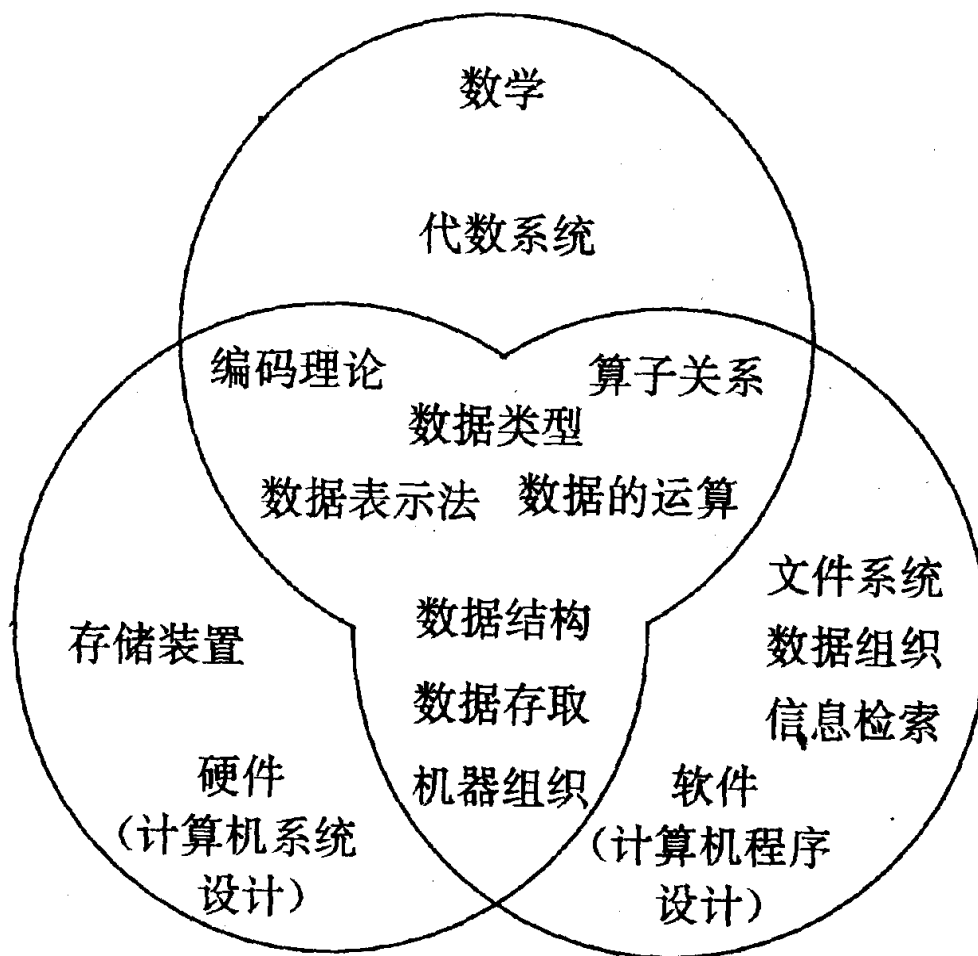
发展阶段:

数据结构的概念不断扩充, 包括了网络、集合代数论、关系等“离散数学结构”的内容。

70年代后期, 我国高校陆续开设该课程。

《数据结构》所处的地位:

介于数学、计算机硬件和计算机软件三者之间的一门核心课程



数据结构在计算机学科中的地位

Web信息处理

队列、图、字符、矩阵、哈希表、排序、索引、检索

人工智能

广义表、集合、有向图、搜索树

图形图像

队列、栈、图、矩阵、空间索引树、检索

数据库

线性表、多链表、排序、B+树

操作系统

队列、存储管理表、排序、目录树

编译原理

字符串、栈、哈希表、语法树

算法分析与设计

数据结构与算法

计算复杂性理论

程序设计语言

概率统计

计算概论

集合论与图论

课程目的

- 能够分析研究计算机加工的对象特性，获得其逻辑结构，根据需求，选择合适存储结构及其相应的算法；
- 学习一些常用的算法；
- 复杂程序设计的训练过程，要求编写的程序结构清楚和正确易读；
- 初步掌握算法的时间分析和空间分析技术



1.2 基本概念和术语

- 1、**数据** (data)—所有能输入到计算机中去的描述客观事物的符号
 - ◆ 数值性数据
 - ◆ 非数值性数据 (多媒体信息处理)
- 2、**数据元素** (data element) —数据的基本单位, 也称结点 (node) 或记录 (record)
- 3、**数据项** (data item) —有独立含义的数据最小单位, 也称域(field)

三者之间的关系：数据 > 数据元素 > 数据项

例：学生表 > 个人记录 > 学号、姓名……

4、数据对象(Data Object): 相同特性数据元素的集合, 是数据的一个子集

- ◆ 整数数据对象

$$\mathbb{N} = \{ 0, \pm 1, \pm 2, \dots \}$$

- ◆ 学生数据对象

- 学生记录的集合

5、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。

数据结构是带“结构”的数据元素的集合，
“结构”就是指数据元素之间存在的关系。

数据结构 的两个层次:

逻辑结构---

数据元素间抽象化的相互关系，与数据的存储无关，独立于计算机，它是从具体问题抽象出来的数学模型。

存储结构（物理结构） ----

数据元素及其关系在计算机存储器中的存储方式。

划分方法一

(1) 线性结构----

有且仅有一个开始和一个终端结点，并且所有结点都最多只有一个直接前趋和一个后继。

例如：线性表、栈、队列、串

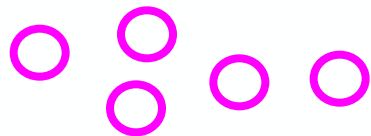
(2) 非线性结构----

一个结点可能有多个直接前趋和直接后继。

例如：树、图

划分方法二

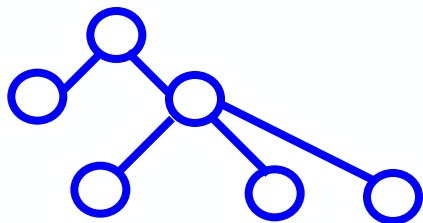
集合——数据元素间除“同属于一个集合”外，无其它关系



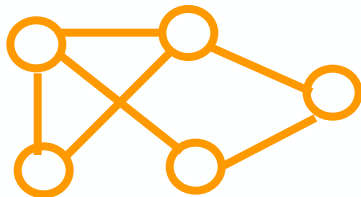
线性结构——一个对一个，如线性表、栈、队列 (幼儿园排队)



树形结构——一个对多个，如树



图形结构——多个对多个，如图



存储结构分为：

顺序存储结构——借助元素在存储器中的**相对位置**来表示
数据元素间的逻辑关系

[Click Here](#)

链式存储结构——借助指示元素存储地址的**指针**表示数据
元素间的逻辑关系

[Click Here](#)

链式存储: 银行排队, 潜伏

[Back](#)

顺序存储

存储地址	存储内容
L_0	元素1
L_0+m	元素2

$L_0+(i-1)*m$	元素i

$L_0+ (n-1)*m$	元素n

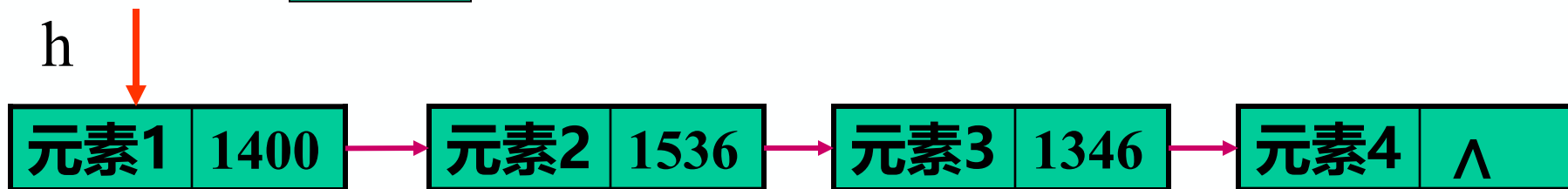
$$\text{Loc(元素i)} = L_0 + (i-1)*m$$

h

链式存储

1345

h



存储地址	存储内容	指针
1345	元素1	1400
1346	元素4	Λ
.....
1400	元素2	1536
.....
1536	元素3	1346

Back

数据的运算

- 逻辑结构和存储结构都相同,但运算不同,则数据结构不同. 例如,栈与队列
- 对于一种数据结构,常见的运算
 - 插入
 - 删除
 - 修改
 - 查找
 - 排序

逻辑结构

唯一

存储结构

不唯一

↑
运算的实现

依赖于

存储结构

数据的逻辑结构

线性结构

线性表

栈、队列

串、数组

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

数据的运算：插入、删除、修改、查找、排序

数据类型

- **定义：** 在一种程序设计语言中，变量所具有的数据种类

FORTRAN语言： 整型、实型、和复数型

C语言：

基本数据类型： char int float double void

构造数据类型： 数组、结构体、共用体、文件

- ✓ 数据类型是一组性质相同的值的集合, 以及定义于这个集合上的一组运算的总称

抽象数据类型

(ADTs: Abstract Data Types)

- ◆ 更高层次的数据抽象
- ◆ 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，并包括一组相关的操作

3D point

超级玛丽

抽象数据类型可以用以下的三元组来表示：

$$\text{ADT} = (\text{D}, \text{S}, \text{P})$$

数据对象 D上的关系集 D上的操作集

ADT
常用
定义
格式

ADT抽象数据类型名{

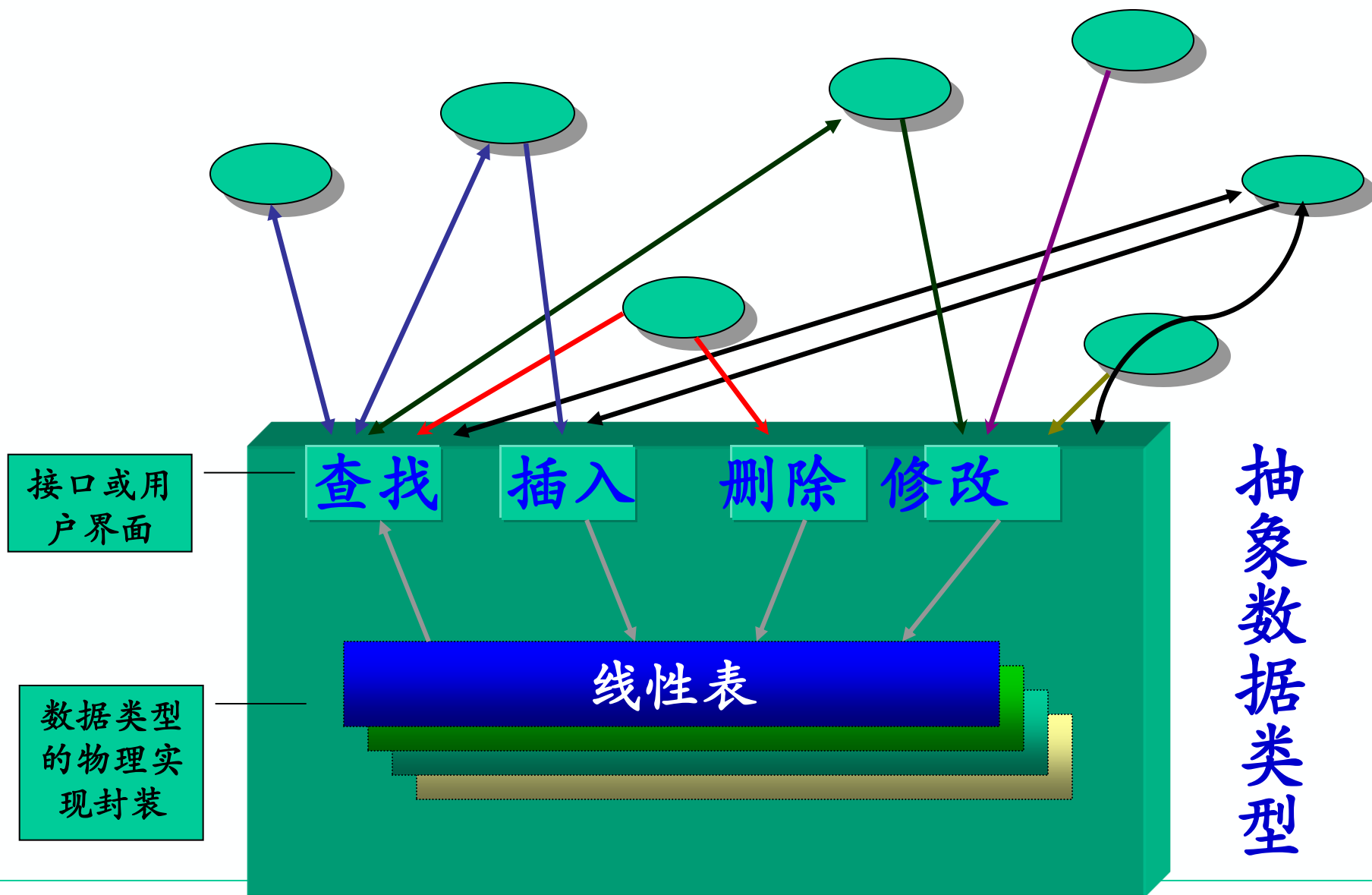
数据对象：<数据对象的定义>

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

信息隐蔽和数据封装，使用与实现相分离



1.3 抽象数据类型的表示与实现



抽象数据类型可以通过**固有的**数据类型（如整型、实型、字符型等）来表示和实现。

它有些类似C语言中的**结构（struct）类型**，但增加了相关的**操作**

教材中用的是**类C语言**（介于伪码和C语言之间）
作为描述工具

但上机时要用具体语言实现，如C或C++等

- (1) 预定义常量及类型

- // 函数结果状态代码

```
#define OK 1
```

```
#define ERROR 0
```

```
#define OVERFLOW -2
```

- // Status是函数返回值类型，其值是函数结果状态代码。

```
typedef int Status;
```


- (2) 数据元素被约定为 ElemType 类型，用户需要根据具体情况，自行定义该数据类型。

(3) 算法描述为以下的函数形式：

函数类型 函数名 (函数参数表)

{

语句序列;

}

(4) 内存的动态分配与释放

使用new和delete动态分配和释放内存空间

分配空间 指针变量=new数据类型;

释放空间 delete指针变量;

(5) 赋值语句

(6) 选择语句

(7) 循环语句

(8) 使用的结束语句形式有:

函数结束语句 `return`

循环结束语句 `break;`

异常结束语句 `exit (异常代码) ;`

(9) 输入输出语句形式有:

输入语句 `cin (scanf())`

输出语句 `cout (printf())`

(10) 扩展函数有:

求最大值 `max`

求最小值 `min`



1.4 算法和算法分析

- 算法定义：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
- 算法的描述：
 - ◆ 自然语言
 - ◆ 流程图
 - ◆ 程序设计语言
 - ◆ 伪码

■ 算法的特性:

- ◆ **输入** 有0个或多个输入
- ◆ **输出** 有一个或多个输出(处理结果)
- ◆ **确定性** 每步定义都是确切、无歧义的
- ◆ **有穷性** 算法应在执行有穷步后结束
- ◆ **有效性** 每一条运算应足够基本

算法的评价

- ◆ 正确性
- ◆ 可读性
- ◆ 健壮性
- ◆ 高效性（时间代价和空间代价）

算法的效率的度量

- 算法效率：用依据该算法编制的程序在计算机上执行所消耗的时间来度量

事后统计

事前分析估计

1.事后统计：利用计算机内的计时功能，不同算法的程序可以用一组或多组相同的统计数据区分

缺点：

- ①必须先运行依据算法编制的程序
- ②所得时间统计量依赖于硬件、软件等环境因素，掩盖算法本身的优劣

2.事前分析估计:

一个高级语言程序在计算机上运行所消耗的时间取决于:

- ①依据的算法选用何种策略
- ②问题的规模
- ③程序语言
- ④编译程序产生机器代码质量
- ⑤机器执行指令速度

同一个算法用不同的语言、不同的编译程序、在不同的计算机上运行，效率均不同，——使用**绝对时间单位**衡量算法效率不合适

时间复杂度的渐进表示法

- 算法中基本语句重复执行的次数是问题规模 n 的某个函数 $f(n)$, 算法的时间复杂度记作:

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

上, 算法执行的时间的增长速率和

称渐进

n 越大算法的执行时间越长

- ◆ 排序: n 为记录数
- ◆ 矩阵: n 为矩阵的阶数
- ◆ 多项式: n 为多项式的项数
- ◆ 集合: n 为元素个数
- ◆ 树: n 为树的结点数
- ◆ 图: n 为图的顶点数或边数

◆ 算法中重复执行次数和算法的执行时间成正比的语句

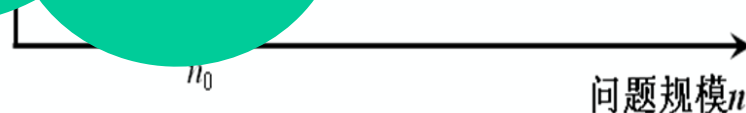
◆ 对算法运行时间的贡献最大

合上 n 数, 则

$T(n) = O(f(n))$ 表示存在正

C 和 n_0 , 使得当 $n \geq n_0$ 时都

$$0 \leq T(n) \leq Cf(n)。$$



$n * n$ 阶矩阵加法:

```
for( i = 0; i < n; i++)
```

```
    for( j = 0; j < n; j++)
```

```
        c[i][j] = a[i][j] + b[i][j];
```

语句的频度 (Frequency Count): 重复执行的次数: $n * n$;

$T(n) = O(n^2)$

即: 矩阵加法的运算量和问题规模的平方是同一个量级

分析算法时间复杂度的基本方法

- 找出语句频度最大的那条语句作为基本语句
- 计算基本语句的频度得到问题规模 n 的某个函数 $f(n)$
- 取其数量级用符号“ O ”表示

```
x = 0; y = 0;
```

```
for ( int k = 0; k < n; k ++ )  
    x ++;
```

```
for ( int i = 0; i < n; i ++ )
```

```
    for ( int j = 0; j < n; j ++ )
```

```
        y ++;
```

$$T(n) = O(n^2)$$

$$f(n) = n^2$$

时间复杂度是由**嵌套最深层语句**的频度决定的


```
void exam ( float x[ ][ ], int m, int n ) {  
    float sum [ ];  
    for ( int i = 0; i < m; i++ ) {  
        sum[i] = 0.0;  
        for ( int j = 0; j < n; j++ )  
            sum[i] += x[i][j]; •  
    }  
    for ( i = 0; i < m; i++ )  
        cout << i << " : " << sum [i] << endl;  
}
```

$T(n) = O(m * n)$

$f(n) = m * n$

例1：N×N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];  
    }
```


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

算法中的基本操作语句为 $c[i][j]=c[i][j]+a[i][k]*b[k][j];$

$$T(n) = \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n 1 = \sum_{i=1}^n \sum_{j=1}^n n = \sum_{i=1}^n n^2 = n^3 = o(n^3)$$

例2:

```
for( i=1; i<=n; i++)  
    for (j=1; j<=i; j++)  
        for (k=1; k<=j; k++)  
            x=x+1;
```

定理1.1

若 $f(n)=a_m n^m+a_{m-1} n^{m-1}+\dots+a_1 n+a_0$ 是 m 次多项式, 则 $T(n)=O(n^m)$ 。

忽略所有低次幂项和最高次幂系数, 体现出增长率的含义

$$\begin{aligned}\text{语句频度} &= \sum_{i=1}^n \sum_{j=1}^i \sum_{k=1}^j 1 = \sum_{i=1}^n \sum_{j=1}^i j = \sum_{i=1}^n \frac{i(i+1)}{2} \\ &= \frac{1}{2} \left(\sum_{i=1}^n i^2 + \sum_{i=1}^n i \right) \\ &= \frac{1}{2} \left(\frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2} \right) \\ &= \frac{n(n+1)(n+2)}{6}\end{aligned}$$

例3：分析以下程序段的时间复杂度

```
i=1; ①
```

```
while(i<=n)
```

```
    i=i*2; ②
```

$$2^{f(n)} \leq n$$

即 $f(n) \leq \log_2 n$, **取最大值** $f(n) = \log_2 n$

所以该程序段的时间复杂度 $T(n) = O(\log_2 n)$

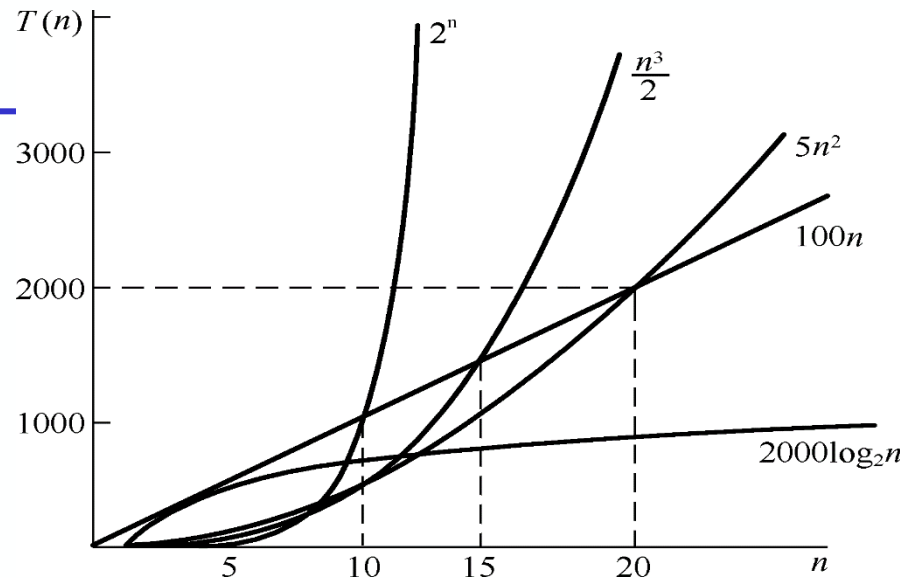
有的情况下，算法中基本操作重复执行的次数还随问题的**输入数据集**不同而不同

例4：顺序查找，在数组 $a[i]$ 中查找值等于 e 的元素，返回其所在位置。

```
for (i=0;i< n;i++)  
    if (a[i]==e) return i+1;  
return 0;
```

- 最好情况：1次
- 最坏情况： n
- 平均时间复杂度为： $O(n)$

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



时间复杂度 $T(n)$ 按数量级递增顺序为：

复杂度低

复杂度高

常数阶	对数阶	线性阶	线性对数阶	平方阶	立方阶	...	K次方阶	指数阶
$O(1)$	$O(\log_2 n)$	$O(n)$	$O(n \log_2 n)$	$O(n^2)$	$O(n^3)$		$O(n^k)$	$O(2^n)$

渐进空间复杂度

- 空间复杂度: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)

■ 算法要占据的空间

- 算法本身要占据的空间, 输入/输出, 指令, 常数, 变量等
- 算法要使用的辅助空间

$S(n) = O(1)$

原地工作

且a中的n个数逆序存放到

 $S(n) = O(n)$

【算法1】

```
for(i=0; i<n/2; i++)  
{  
    t=a[i];  
    a[i]=a[n-i-1];  
    a[n-i-1]=t;  
}
```

【算法2】

```
for(i=0; i<n; i++)  
    b[i]=a[n-i-1];  
for(i=0; i<n; i++)  
    a[i]=b[i];
```

- 1、数据、数据元素、数据项、数据结构等基本概念
- 2、对数据结构的两个层次的理解
 - 逻辑结构
 - 存储结构
- 3、抽象数据类型的表示方法
- 4、算法、算法的时间复杂度及其分析的简易方法（最差复杂度：找东西）