课程名称：数据结构与算法

数据结构

教材名称：

《数据结构教程(第二版)》

唐发根　编著

北京航空航天大学出版社 2005（2012 年4月第7次印刷）

1. 《数据结构(C语言版)》、《数据结构题集(C语言版)》
   清华大学出版社　严蔚敏 吴伟民
2. 《数据结构习题与解析》 科学出版社 2002

为什么要学习数据结构

- 计算机软件与理论学科的专业基础课程
- 后续专业课程学习的必要知识与技能准备
  - 编译技术要使用栈、散列表及语法树
  - 操作系统中用队列、存储管理表及目录树
  - 数据库系统运用线性表、多链表、及索引树
  - etc.
- 增强读者求解复杂问题的能力

目的

通过本课程的学习，运用本课程讨论的知识学会合理地组织数据，有效地表示数据，有效地处理数据，更好地进行算法设计与算法分析，掌握计算机进行数据处理的基本原理、基本方法和技巧，进一步提高程序设计的水平和能力。

本课程的特点：

1. 计算机专业重要的专业基础课之一。
2. 最好有有关“程序设计语言”和“离散数学”的知识作为课程的基础。
3. 实践性较强。

本课程的相关信息：

1. 本课程的网站：http://bhds.buaa.edu.cn
2. 考核方式：闭卷笔试/机考(80%)＋平时成绩（作业＋考勤）（20%)

Email:qhb@buaa.edu.cn 电话：82317629

教学（作业）辅助平台：

http://judge.buaa.edu.cn（校园网）

IP:115.25.138.229(暂时）

辅导老师：夏清（13426016742）李竞雪、、翟骁

3. 考勤+选班联络人
4. 上机安排： 18小时（网络中心老师安排 ）

要求：诚信

- 按时提交作业，上机作业都必须在指定的期限内完成并提交
- 提倡讨论，但严禁抄袭
  - ✓可以讨论思路，请同学看算法的逻辑问题和效率问题。
  - ✓但要亲自动手实现。
- 发现抄袭，作业扣分，教学平台自动检查

第一章 绪论

本章内容

1.1 什么是数据结构

1.2 算法及其描述

1.3 算法分析的基本概念

1.1 什么是数据结构

一. 名词术语

数据

描述客观事物的数字、字符以及一切能够输入到计算机中，并且能够被计算机程序处理的**符号的集合** 。

数据元素

数据这个集合中的一个一个的元素。

数据对象

具有相同特性的数据元素的**集合**。

结 构

数据元素之间具有的关系。

数 据

数据元素

1

(25, 78, 36, 100, 28, 45)

数列

2

(‘A’, ‘B’, ‘C’, …, ‘Z’)

字母表

3

学 号	姓 名	性别	年龄	其 他
99001	张 三	女	17	……
99002	李 四	男	16	……
99003	王 五	女	18	……
99004	周 六	女	17	……
⋮	⋮	⋮	⋮	⋮
99035	刘 末	男	19	……

数据文件

二. 数据结构的定义

1. 数据元素之间的联系称之为 **结构**，**数据结构**就是具有结构的数据元素的集合。

2. **数据结构**是一个二元组

Data-Structure=(D, R)

其中，**D**是数据元素的有限集合，**R**是**D**上的关系的集合。

某一数据对象

逻辑关系

逻辑结构

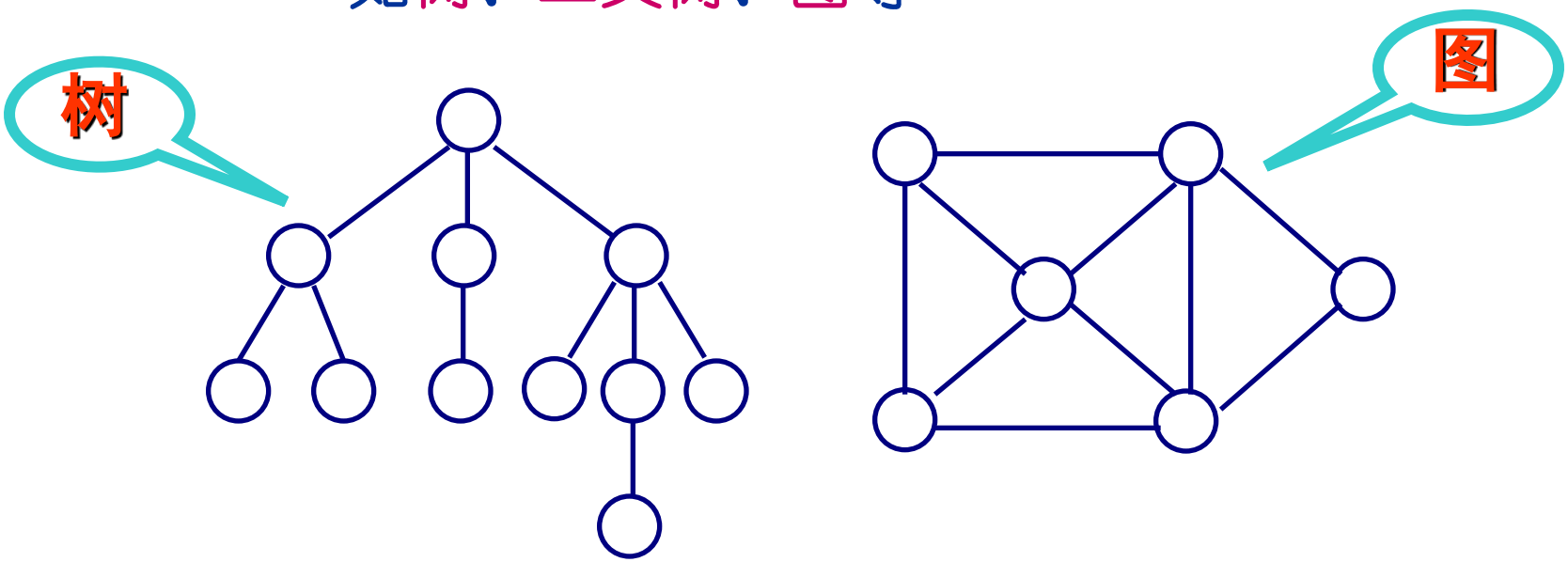
数据元素之间具有的逻辑关系(结构)。

线性关系

如线性表、堆栈、队列、串、文件等

非线性关系

如树、二叉树、图等



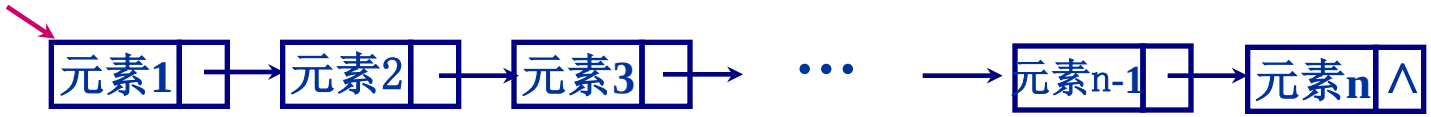
存储结构　具有某种逻辑结构的数据在计算机存储器中的存储方式(存储映象)。

1. 顺序存储结构

用一组**地址连续**的存储单元依次存放数据元素，数据元素之间的逻辑关系通过元素的地址直接反映。

2. 链式存储结构

用一组**地址任意**的存储单元依次存放数据元素，数据元素之间的逻辑关系通过指针间接地反映。



链表结构！

3. 索引存储结构

构造原理:

利用数据元素的索引关系来确定数据元素的存储位置，由数据元素本身与索引表两部分组成。

特点:

诸如查找、插入和删除等操作的时间效率较高，但存储空间开销较大。

第九章详细讨论

4. 散列存储结构

构造原理:

通过事先准备好的散列函数关系与处理冲突的方法来确定数据元素的存储位置。

特点:

诸如查找、插入和删除等操作的时间效率较高,主要缺点是确定好的散列关系比较困难。

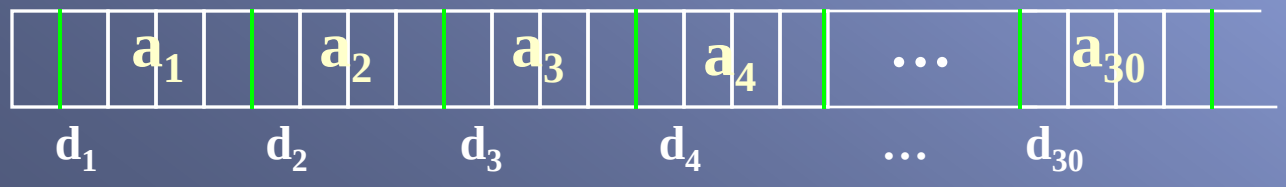
例

	姓 名	性别	民族	年龄	其 他
a_1	刘晓光	男	汉	16	…
a_2	马广生	男	回	17	…
a_3	王 民	男	壮	19	…
⋮	⋮	⋮	⋮	⋮	⋮
a_{30}	张淑华	女	汉	24	…

逻辑结构： 线性结构（线性表）

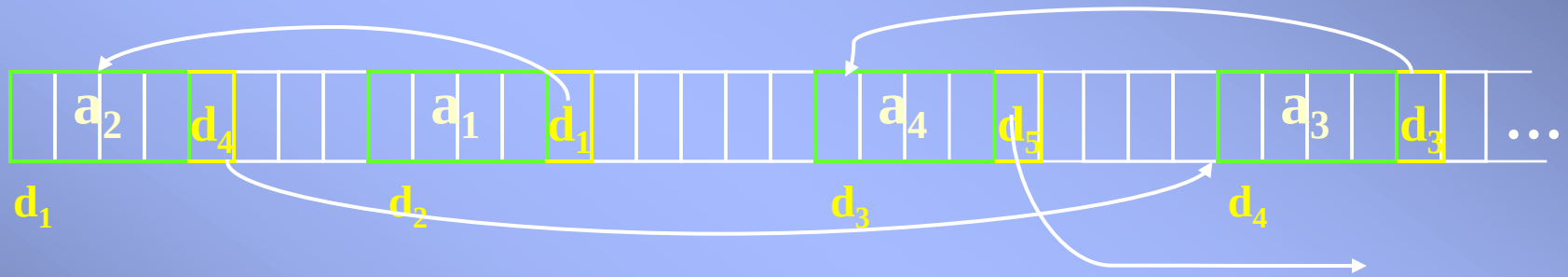
存储结构：

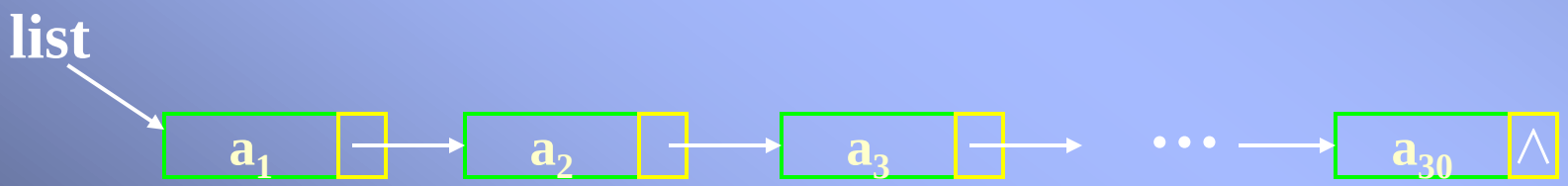
1. 顺序存储结构



2. 链式存储结构

用一片地址任意的存储空间！





链表

三. 数据结构课程研究的主要内容

逻辑结构

1. 研究数据元素之间的客观联系。
2. 研究具有某种逻辑关系的数据在计算机存储器内的存储方式。

存储结构

3. 研究如何在数据的各种关系(逻辑的和物理的)的基础上对数据实施一系列有效的基本操作。

算法

逻辑结构 + 存储结构 + 算法

1.2 算法及其描述

一. 算法及其性质

1. 算法的定义

算法不是问题解决的答案，而是获得答案的过程。

(1). **算法**是用来解决某个特定课题的指令的集合。

(2). **算法**是由人们组织起来准备加以实施的一系列有限的基本步骤。

(3). **算法**是一组解决问题的清晰指令，它能够对符合一定规范的输入，在有限的时间内获得所需要的输出。

2. 算法的性质

一个完整的算法应该满足下面五个基本标准：

输入 由算法的外部提供$n \geq 0$个有限量作为算法的输入。

输出 由算法的内部提供$n > 0$个有限量作为算法的输出。

有穷性 算法必须在有限的步骤内能够结束。

确定性 组成算法的每一条指令必须有清晰明确的含义。

有效性 算法的每一条指令必须具有可执行性。

二、算法的描述

1. 采用自然语言来描述

问题： 求两个正整数M与N的最大公因子。

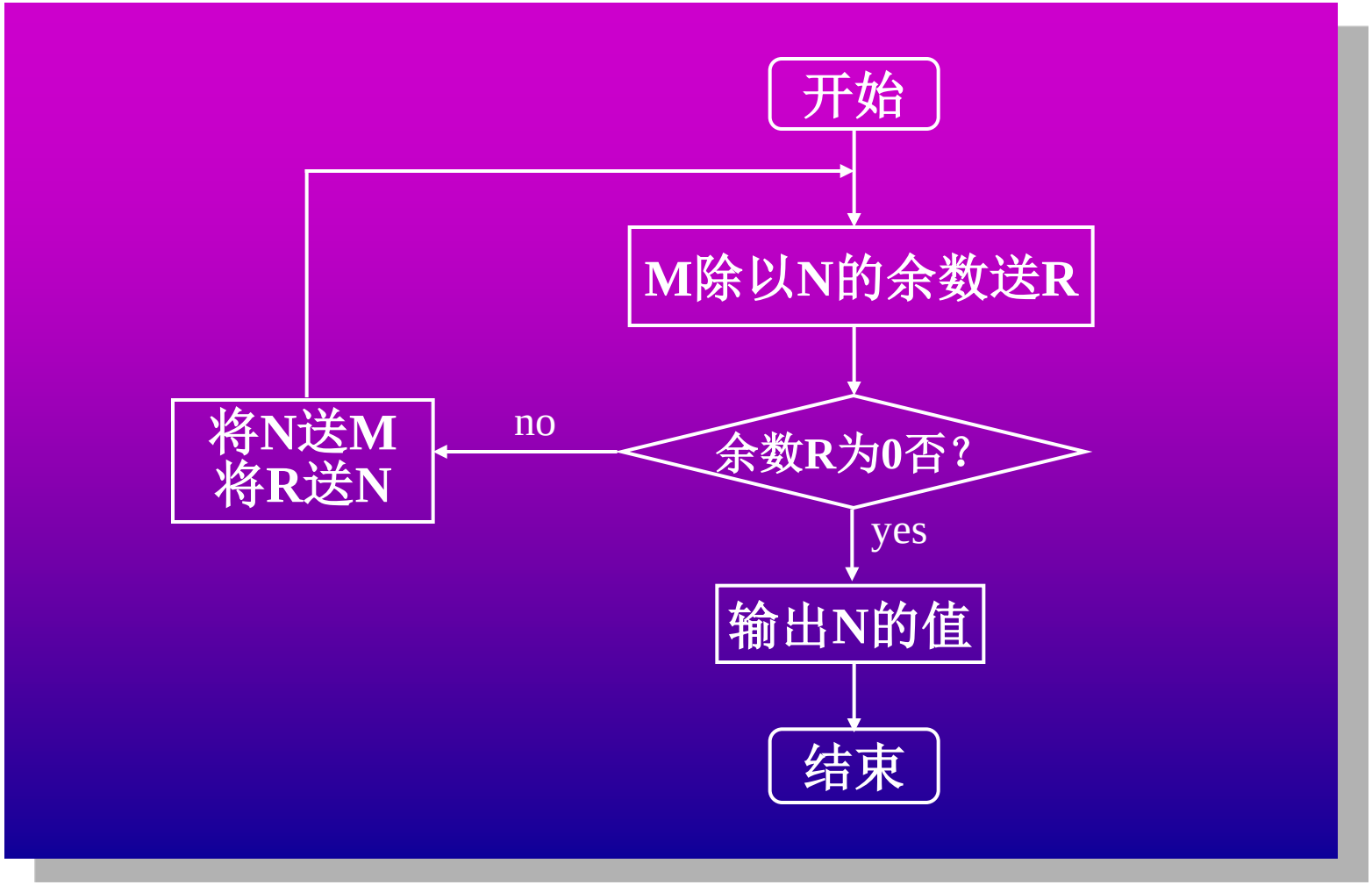
(1) M除以N，将余数送中间变量R；

(2) 测试余数R是否等于零？

a) 若R等于零，求得的最大公因子为当前N的值, 算法到此结束。

b) 若R不等于零，将N送入M，将R送N, 重复算法的(1)和(2)。

2. 采用程序流程图的形式来描述



3. 设计一种既脱离某种具体的程序设计语言，又具有各种程序设计语言的共同特点的形式化语言来描述。

类Pascal语言　　类C语言

SPARKS语言，一种类Pascal语言

4. 采用某种具体程序语言来描述

Pascal　C、C++、JAVA

```
int COMFACTOR( int M, intN )
    {
        int  R;
        while ( 1 )
            {
              R=M  %  N;
              if  (R==0)
                   return  N;
              M=N;
              N=R;
            }
    }
```

用C语言描述的求两个正整数最大公因子的算法(C函数)

本书算法涉及到的C语言成分

算法格式

函数

```
函数类型  函数名(参数表)
{
        语句序列
}
```

算法名

包括说明、执行语句

赋值语句

简单赋值

变量名=表达式;

条件赋值

变量名=条件表达式？表达式(真):表达式(假);

条件语句

条件语句1

```
if(表达式)
    语句序列;
```

条件语句2

```
if(表达式)
    语句序列1;
else
    语句序列2;
```

开关语句

开关语句1

```
switch(表达式){
    case 值1: 语句序列1; break;
      …
    case 值n: 语句序列n; break;
    default:  语句序列n+1;
}
```

开关语句2

```
switch{
    case 条件1: 语句序列1; break;
      …
    case 条件n: 语句序列n; break;
    default:  语句序列n+1;
}
```

循环语句

for语句

```
for(赋值表达式序列;循环条件;修改表达式序列)
    语句序列;
```

while语句

```
while(条件)
    语句序列;
```

do_while语句

```
do{
    语句序列;
}while(条件);
```

结束语句

函数返回语句	return 表达式;
case结束语句	break;
异常结束语句	exit(异常代码);

输入/输出语句

输入语句	scanf([格式串], 变量1, …, 变量n);
输出语句	printf([格式串], 表达式1, …, 表达式n);

注释行

```
/* 注释内容 */
```

逻辑运算的约定

与运算&&

对于A&&B, 当A的值为假时, 不再对B求值。

或运算 ||

对于A||B, 当A的值为真时, 不再对B求值。

1.3 算法分析

目的：改进算法的质量

前提：被分析的算法必须正确

算法分析是指对算法质量优劣的评价。

除正确性外，应从三个方面分析一个算法：

- 依据算法编写的程序在计算机中运行时间多少的度量，称之为***时间复杂度***。

  反映算法运行的快慢

- 依据算法编写的程序在计算机中占存储空间多少的度量，称之为***空间复杂度***

  反映算法需要额外空间的多少

- 其他方面。如算法的可读性、可移植性以及易测试性的好坏。

时空效率高的算法才是一个好的算法，它常常是不懈努力和反复修正的结果

时间复杂度

一个程序在计算机中运行时间的多少与诸多因素有关，其中主要有：

相关

1. 问题的规模。

几乎所有算法的时间效率都与问题的规模有关

2. 编译程序功能的强弱以及所产生的机器代码质量的优劣。

3. 机器执行一条指令的时间长短。

4. 程序中那些关键语句的执行次数。

重点

对算法运行时间贡献最大的语句

频度统计法

以语句执行的次数的多少作为算法的时间量度的分析方法称为**频度统计法**。

一条**语句的频度**是指该语句被执行的次数，而整个**算法的频度**是指算法中所有语句的频度之和。

例1

```
void MATRIX( int A[ ][n], int B[ ][n], int C[ ][n], int n )
{
    int i, j, k;
    for(i=0;i<n;i++)                          ---------- n+1
        for(j=0;j<n;j++){                     ---------- n(n+1)
            C[i][j]=0;                        ---------- n²
            for(k=0;k<n;k++)                  ---------- n²(n+1)
                C[i][j]=C[i][j]+A[i][k]*B[k][j];   -------- n³
        }
}
```

求两个n阶矩阵的乘积

算法的频度：

$$f(n) = n+1 + n(n+1) + n^2 + n^2(n+1) + n^3$$
$$= 2n^3 + 3n^2 + 2n + 1$$

关于符号O的定义

当且仅当存在正整数c和n_0，使得$f(n) \le cg(n)$对所有的$n \ge n_0$成立，则称函数f(n)与g(n)同阶，或者说，f(n)与g(n)同一个数量级，记作

$$f(n)=O(g(n))$$

称上式为算法的时间复杂度，或称该算法的时间复杂度为O(g(n))。其中，n为问题的规模（大小）的量度。

当问题的输入规模n趋于无穷大时，算法的运行时间表现出固定的增长次数。

$f(n)=2n^3+3n^2+2n+1$　　$g(n)=n^3$

称算法的时间复杂度为$O(n^3)$。

例2

```
i=1;                    (1)
While (i<=n)            (2)
  i=i*2;                (3)
```

语句(1)的频度为1

设语句(3)的频度为t(n),则有:

$2^{t(n)}<=n$

即$t(n)<=\log_2 n$, 取最大值$t(n)=\log_2 n$

语句(2)的频度为t(n)+1, 即 $\log_2 n+1$

算法的频度：

$$f(n) = 1+t(n)+1+t(n)=2+ 2\log_2 n$$

时间复杂度为$O(\log_2 n)$

例3

求如下递归函数 fact(n), 分析其时间复杂度。

```
Fact(int n)
{
   if (n<=1 ) return(1);        (1)
   else return(n*fact(n-1)); (2)
}
```

设该函数的运行时间函数是T(n),语句(1)的运行时间是O(1)

语句(2)的运行时间是T(n-1)+O(1)，其中O(1)为运算的时间。

因此有

$$T(n)=\begin{cases} O(1) & n<=1 \\ T(n-1)+O(1) & n>1 \end{cases}$$

$T(n)=O(1)+T(n-1)=2*O(1)+T(n-2)=\ldots=n*O(1)=O(n)$

即fact(n)的时间复杂度为O(n)。

对于算法分析具有重要意义的常见函数值

n	$\log_2 n$	n	$n\log_2 n$	n^2	n^3	2^n	n!
10	3.3	10^1	3.3×10^1	10^2	10^3	10^3	3.6×10^6
100	6.6	10^2	6.6×10^2	10^4	10^6	1.3×10^{30}	9.3×10^{157}
1000	10	10^3	1.0×10^4	10^6	10^9		
10000	13	10^4	1.3×10^5	10^8	10^{12}		
…	……						

几种常见的复杂度的形式

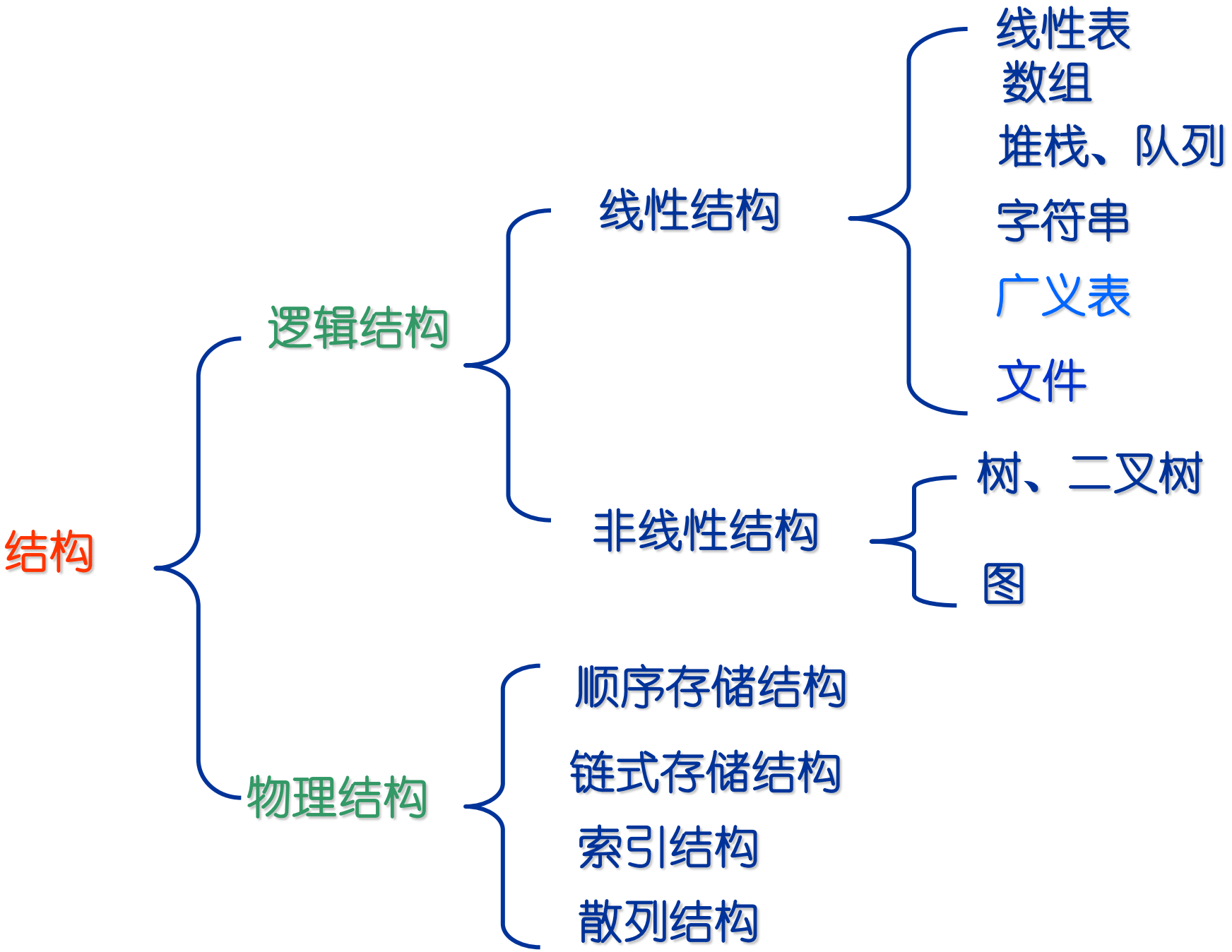
$$O(\log_2 n) < O(n) < O(n\log_2 n) < O(n^2) < O(n^3) < O(2^n)$$

O(1) —— 表示算法的复杂度为常量，不随问题规模n的大小而改变。

$O(\log_2 n)$:对数时间　　O(n):线性时间

$O(n^2)$:平方时间　　$O(n^3)$:立方时间　　$O(2^n)$:指数时间

本章内容小结


结构
逻辑结构
线性结构
线性表
数组
堆栈、队列
字符串
广义表
文件
非线性结构
树、二叉树
图
物理结构
顺序存储结构
链式存储结构
索引结构
散列结构

算法

算法的定义

- 用来解决某个特定课题的指令的集合。
- 由人们组织起来准备加以实施的一系 列有限的基本步骤。

算法的特征

输入、输出、有穷性、确定性、有效性

算法的描述

- 可以采用自然语言或程序设计框图的形式
- 可以采用某一种自定义的符号语言
- 可以采用某一种具体的程序设计语言

算法分析

- 什么是算法分析？
- 算法分析的目的是什么？
- 算法分析的前提是什么？
- 通常从哪几个方面对算法进行分析？