数据结构与算法

宋国杰

北京大学，智能学院

《数据结构与算法》

- **逻辑结构**
  - 图 ⊇ 树 ⊇ 线性
- **存储结构**
  - 顺序方法、链接方法
  - 索引方法、散列方法
- **运算**
  - 增、删、改等简单运算
  - 遍历、排序等复杂运算

时间复杂性　　空间复杂性

课程目标

- 掌握常用的基本数据结构及其应用
  - 包括线性表、栈、队列、串、数组、广义表、树、二叉树、图等基本逻辑结构及其存储结构和所施加的运算
- 学会合理组织数据、有效表示数据和有效处理数据
- 掌握基本的算法设计与分析技术
  - 分析算法的时间复杂性和空间复杂性
- 提高程序设计的质量

课程内容

第1章 概论

第2章 线性表

第3章 栈和队列

第4章 字符串

第5章 二叉树

第6章 树

第7章图

第8章 内排序

第9章 外排序

第10章 检索

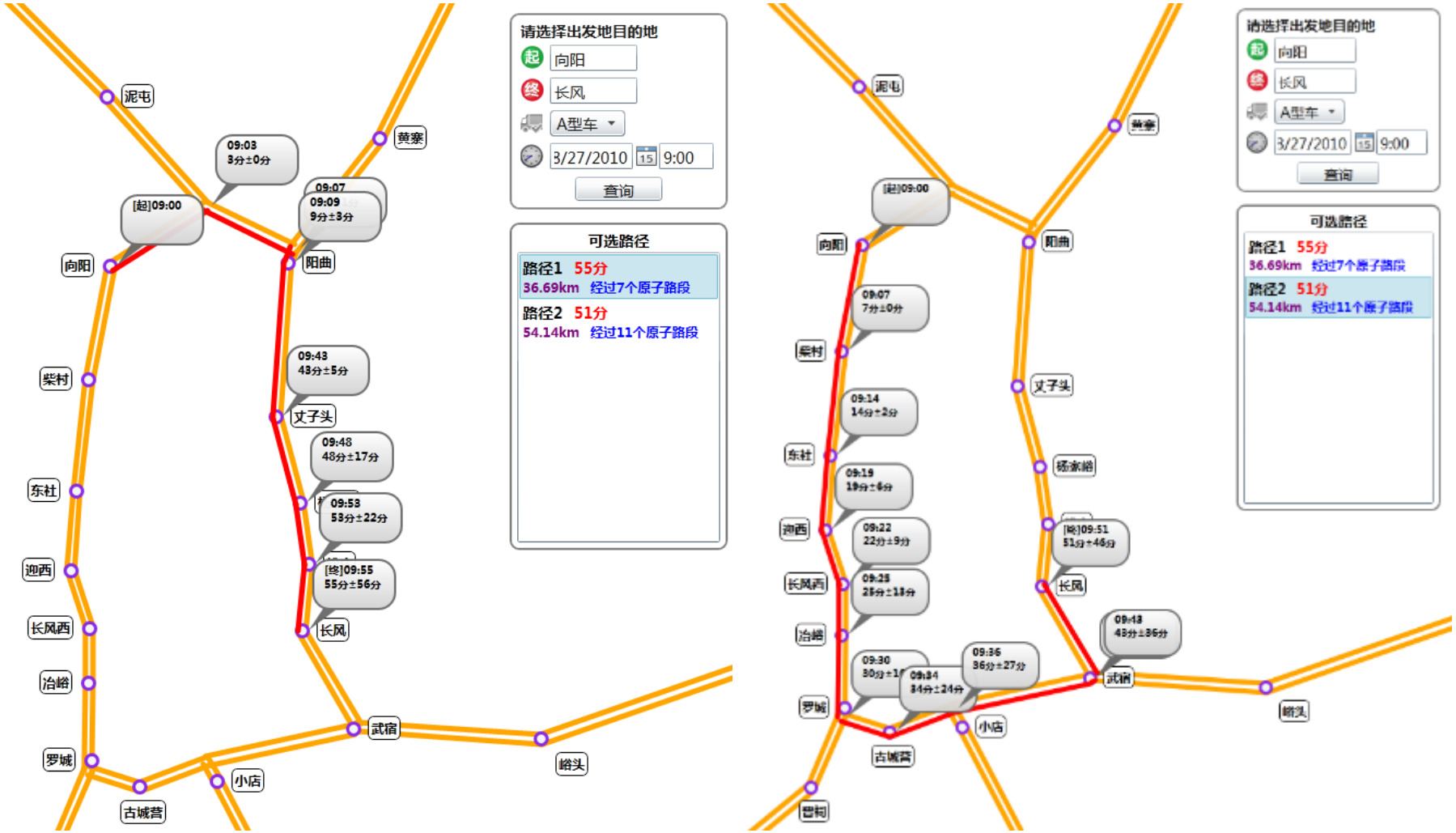
第11章 索引

第12章 高级数据结构

解决问题的过程

- **问题（Problem）**
  - 从输入到输出的一种函数映射
- **数据结构（Data Structure）**
  - 数据的逻辑结构及在计算机中的存储表示，以及相应操作
- **算法（Algorithm）**
  - 对特定问题求解过程的描述方法
- **程序（Program）**
  - 算法在计算机程序设计语言中的实现

出行路径导航问题


泥屯
09:03
3分±0分
黄寨
09:07
09:09
9分±3分
[起]09:00
向阳
阳曲
09:43
43分±5分
柴村
丈子头
09:48
48分±17分
东杜
09:53
53分±22分
迎西
[终]09:55
55分±56分
长风西
长风
冶峪
武宿
罗城
小店
峪头
古城营
请选择出发地目的地
起
向阳
终
长风
A型车
3/27/2010
9:00
查询
可选路径
路径1 55分
36.69km 经过7个原子路段
路径2 51分
54.14km 经过11个原子路段

课程地位


Web信息处理
队列、图、字符、矩阵散列、排序、索引、检索
人工智能
广义表、集合、搜索树及各种有向图
图形图像
队列、栈、图、矩阵、空间索引树、检索
数据库概论
线性表、多链表、排序及B+索引树
操作系统
队列、存储管理表、排序及目录树
编译原理
字符串、栈、散列表及语法树
算法分析与设计
数据结构与算法
概率统计
程序设计实习
集合论与图论
计算概论

课程特点

- 基础性
- 理论性
- 实践性
- 挑战性

教学目标

- **数据的组织与分析能力**
  - 合理的组织数据、表示数据和处理数据
- **抽象能力**
  - 问题->数据->算法
- **算法的设计与分析能力**
  - 操纵数据结构，达到求解问题的目标
- **编程能力**
  - 设计出高质量的程序

教学要求

- 课程讲授
- 诚信作业
- 加强训练
- 有效反馈

书面作业提交要求

- 写学号、名字
- 每次作业，都在纯文本（或者PDF）中写上“**我保证没有抄袭别人的作业**”的诚实保证。否则，记零分或者根据抄袭情况扣分。

诚信

- 端正学习态度、调动学习兴趣
  - 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，则抄袭者和被抄袭者本次作业或上机题期末总评总分扣3分/每次

课程评估

- **平时成绩（40分）**
  - MOOC+POJ 作业（ 15+10 =25 分）
  - 书面作业（+考勤等）（ 15分）
- **考试成绩（60分）**
  - 书面考试：期中（20分）＋期末（25分）
  - 上机考试：POJ（15分）

上机考试

- **时间**：期中笔试之后，某个周三3-4节，随堂考试
- **地点**：计算中心机房 + 信科学院机房
- **时长**：2.5小时
- **题量**：5 道编程题，每题100分，总计500分。每题均有部分分
- **语言**：POJ支持的语言均可，但所有语言的时间与内存限制一样；提供C++代码包

教材

- 张铭，王腾蛟，赵海燕，《数据结构与算法》，高等教育出版社，2008年 6月。——普通高等教育“十一五”国家级规划教材

- 张铭、赵海燕、王腾蛟，《数据结构与算法 - - 学习指导与习题解析》，高等教育出版社，2005年 10月。——“十五”国家级规划教材配套参考书

课程网站

- http://course.pku.edu.cn/webapps/login/

中国大学MOOC

- https://www.icourse163.org/course/PKU-1002534001

注册账号要求


个人信息
密码管理
真实姓名：
XXX [宋老师2017]
真实姓名将显示在证书上哦
Last name：
First name：
头像：
上传新头像

助教信息

姓名	电话	邮件
陈　沁	13548537925 （课程总助教）	chenqink@pku.edu.cn
任远怡	13552695399	2301213304@stu.pku.edu.cn

课程微信群[22-23]

群聊：数算课程微信群（23-24）

第一章 概论

(一). 数据结构

- **数据结构(Data Structure)**
  - **涉及数据之间的逻辑关系、数据在计算机中的存储表示和在这种结构上一组能执行的操作（运算）**
- **三要素**
  - **逻辑结构**
  - **存储（物理）结构**
  - **运算**

逻辑结构

- **逻辑结构 (Logical Structure)**
  - 具体问题的数学抽象，反映事物组成和逻辑关系，并不涉及物理细节与实现
- **逻辑结构的表示**
  - 用一组数据（表示为结点集合K），及数据间的二元关系（关系集合R）表示：

    (K，R)
    - K由有限个结点组成，代表一组有明确结构的数据集
    - R是定义在集合K上的一组关系，每个关系 $r \in R$都是 $K \times K$上的二元关系，描述结点间的逻辑关系

举例

家族成员数据结构

- 每个成员个体作为结点（实体），全部成员组成结点集K
- 家族中各类亲属关系就是一组关系R
  - 其中如母子关系r、兄弟关系r*、和妯娌关系r'等

一个家族的结构

节点的类型

- 基本数据类型
  - 整数类型(integer)、实数类型(real) 、字符类型(char)和指针类型(pointer)
- 复合数据类型
  - 由基本数据类型组合而成的数据结构类型
    - 如数组、结构类型等
  - 复合数据类型本身，又可参与定义更为复杂的结点类型
- 结点的类型可以根据应用的需要来灵活定义

结构(关系)类型

- 逻辑结构（K,R）的分类，讨论重点在关系集R上
- 用R的性质来刻划数据结构的特点，并进行分类
  - 集合结构（set structure）
  - 线性结构（linear structure）
  - 树型结构（tree structure）
  - 图结构（graph structure）

线性结构

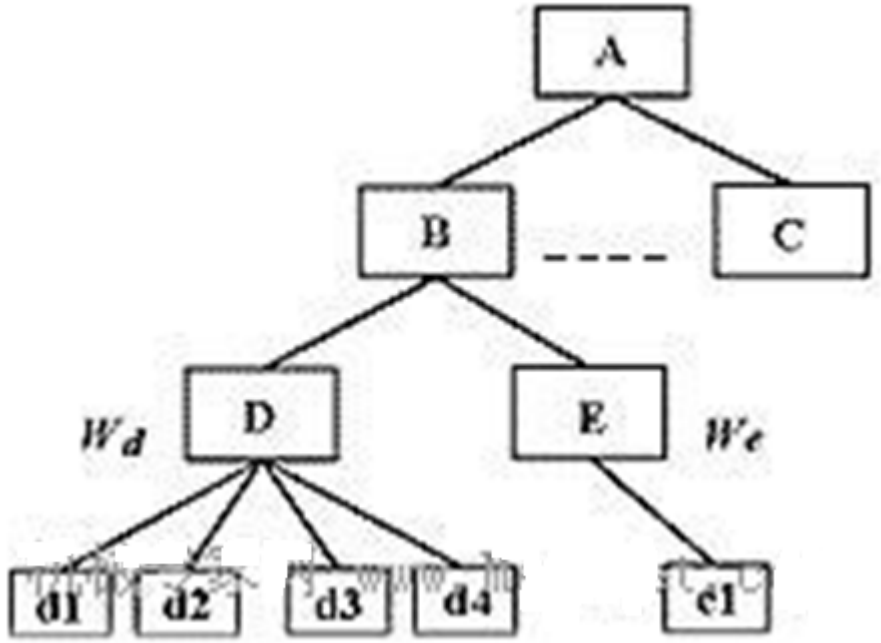
- 亦称 ‘前驱关系’
- 关系r是有向的，满足全序性和单索性约束
  - 全序性：全部结点两两皆可比较前后
  - 单索性：每个结点都存在唯一一个前驱和后继结点

0	1	2	3	4	5	…	N-2	N-1	N

树型结构

- 亦称层次结构，每个结点可有多于1个‘直接后继’，但只有唯一的‘直接前驱’

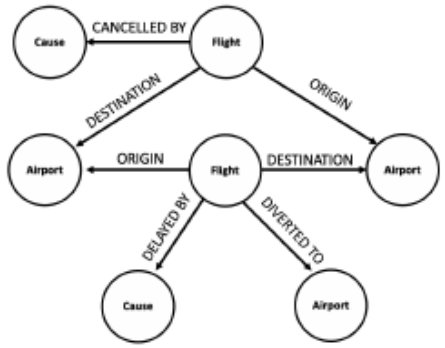
- 最高层结点称为根（root）结点，无父结点



图型结构

- 图结构 亦称 网络结构
  - 交通网、因特网、社会网络等
- 图结构对关系r没有施加任何约束
- 树型结构和图型结构的区别是
  - 每个结点是否仅仅属于一个直接前驱
- 线性结构和树型结构的区别是
  - 每个结点是否仅仅有一个直接后继

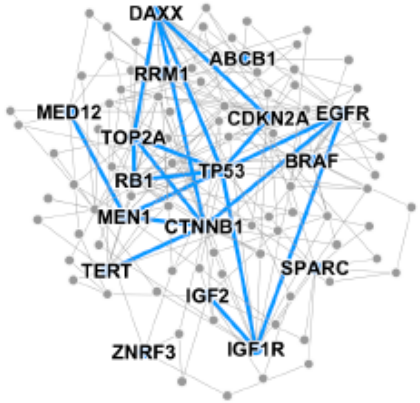
图结构示例



Event Graphs

Image credit: SalientNetworks

Computer Networks



Disease Pathways

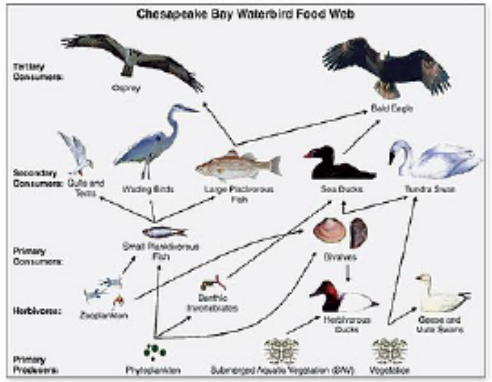


Image credit: Wikipedia

Food Webs

Image credit: Pinterest

Particle Networks

Image credit: visitlondon.com

Underground Networks

图结构示例

Image credit: Medium

Social Networks

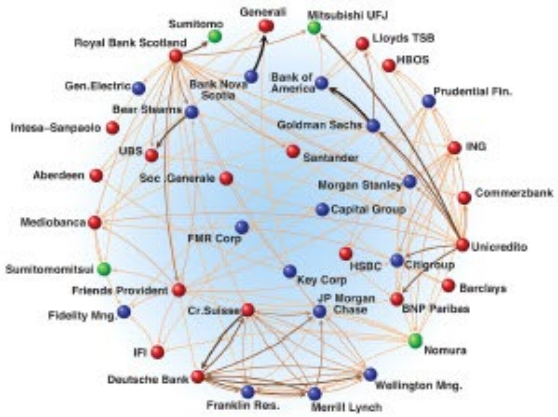


Image credit: Science

Economic Networks

Image credit: Lumen Learning

Communication Networks

Citation Networks

Image credit: Missoula Current News

Internet

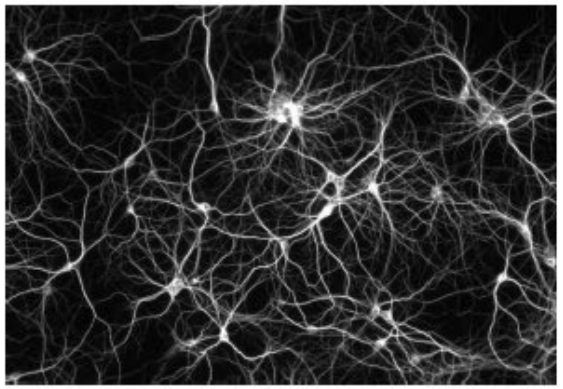


Image credit: The Conversation

Networks of Neurons

图结构示例

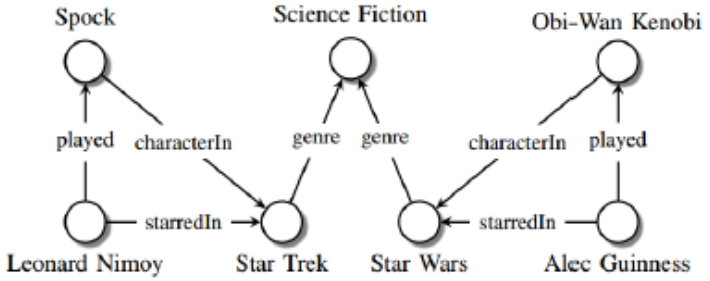


Image credit: Maximilian Nickel et al

Knowledge Graphs

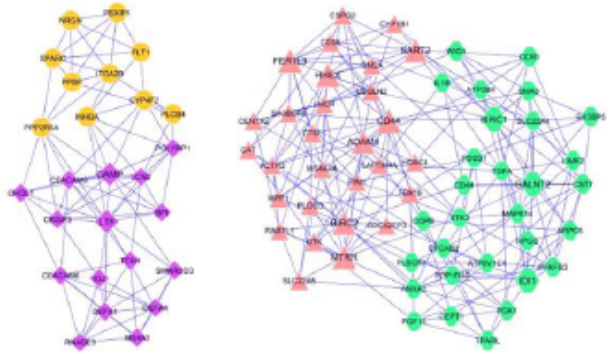


Image credit: ese.wustl.edu

Regulatory Networks

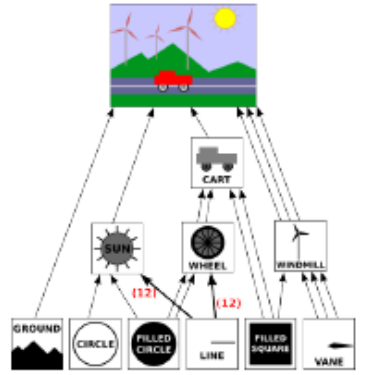


Image credit: math.hws.edu

Scene Graphs

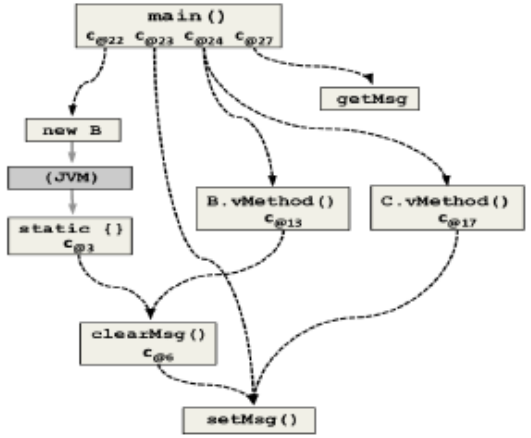


Image credit: ResearchGate

Code Graphs

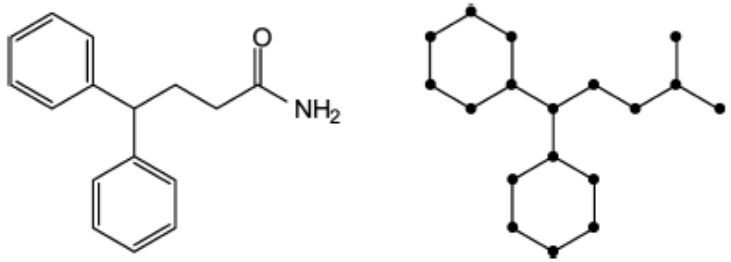


Image credit: MDPI

Molecules

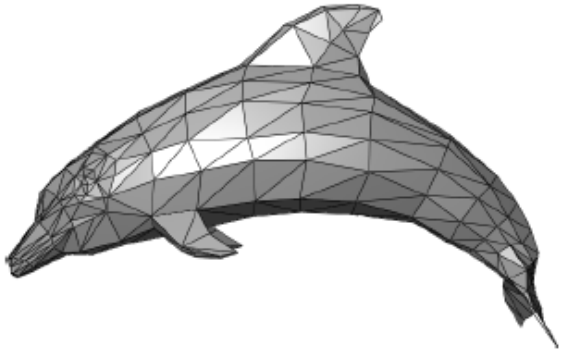


Image credit: Wikipedia

3D Shapes

逻辑结构设计方法

自顶向下的设计逻辑

- 先明确数据结点，及其主要关系r
- 在分析关系r时，也要分析其数据结点的数据类型
- 如果数据结点的逻辑结构比较复杂，那么把它作为下一个层次，再分析下一层次的逻辑结构
- 递归分析，直至满足需求

存储结构

- 数据的存储结构（Storage Structure）
  - 亦称物理结构，是逻辑结构在计算机中的物理存储表示
- 计算机主存储器
  - **空间相邻**：存储空间是一种具有非负整数地址编码的、空间相邻的单元集合，其基本存储单元是字节
  - **随机访问**：计算机指令具有按地址随机访问存储空间内任意单元的能力，访问不同地址所需时间相同

存储结构(Con.)

- 存储结构建立一种**逻辑结构到物理结构的映射**
  - **结点映射**：对于每一个结点$j \in K$都对应一个唯一的存储区域$c \in M$。
  - **关系映射**：每一关系元组（j_1, j_2）$\in r$（其中$j_1, j_2 \in K$是结点），亦即j_1, j_2的逻辑关系映射为存储单元的地址顺序关系（或指针地址指向关系）
- 四种基本存储映射方法
  - 顺序、链接、索引、散列

1、顺序方法

➢ 顺序存储

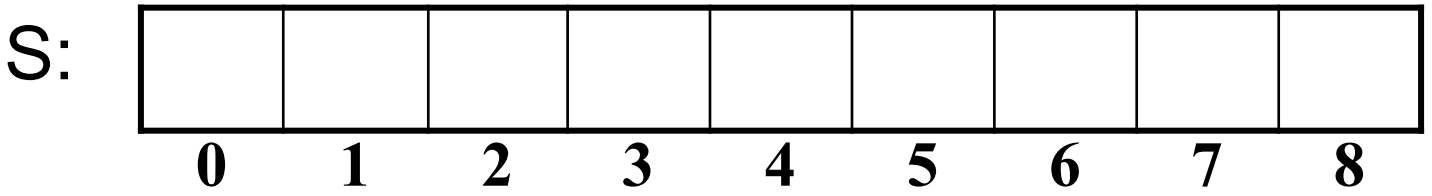
- 结点按地址相邻关系顺序存储，结点间逻辑关系用存储单元的自然顺序关系来表达
- 数组是顺序存储法的具体实例

➢ 顺序存储是紧凑存储结构

- 紧凑指存储空间除存储有用数据外，不存储其他附加信息
- 存储密度：存储结构所存储‘有用数据’和该结构（包括附加信息）整个存储空间大小之比

举例

- 顺序存储结构：一维数组



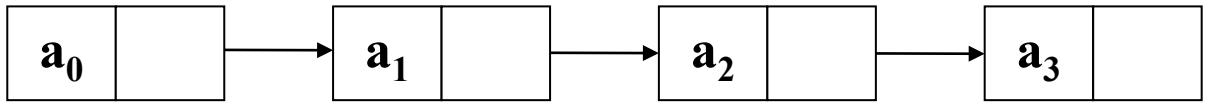
- 二维数组：支持整数编码访问

M_{00}	M_{01}	M_{02}
M_{10}	M_{11}	M_{12}
M_{20}	M_{21}	M_{22}

$$M[i][j]=M[0][0]+(k*i+j)*(\text{元素尺寸})$$

2、链接方法

- 链接法
  - 利用存储结构中指针指向来表示两个结点间逻辑关系
- 可以表达任意的逻辑关系r，将数据结点分为两部分：
  - 数据字段：存放结点本身的数据
  - 指针字段：存放指针，链接到某个后继结点，指向存储单元的开始地址
  - 多个相关结点的依次链接就会形成链表



链接方法(Con.)

- 优点：增删容易
  - 顺序方法对于经常增、删结点情形，往往遇到困难
  - 链接方法结合动态存储可以解决这些复杂的问题

- 缺点：定位困难
  - 访问结点必须知道该结点的指针
  - 否则需要沿着链接指针逐个搜索，花费时间较大

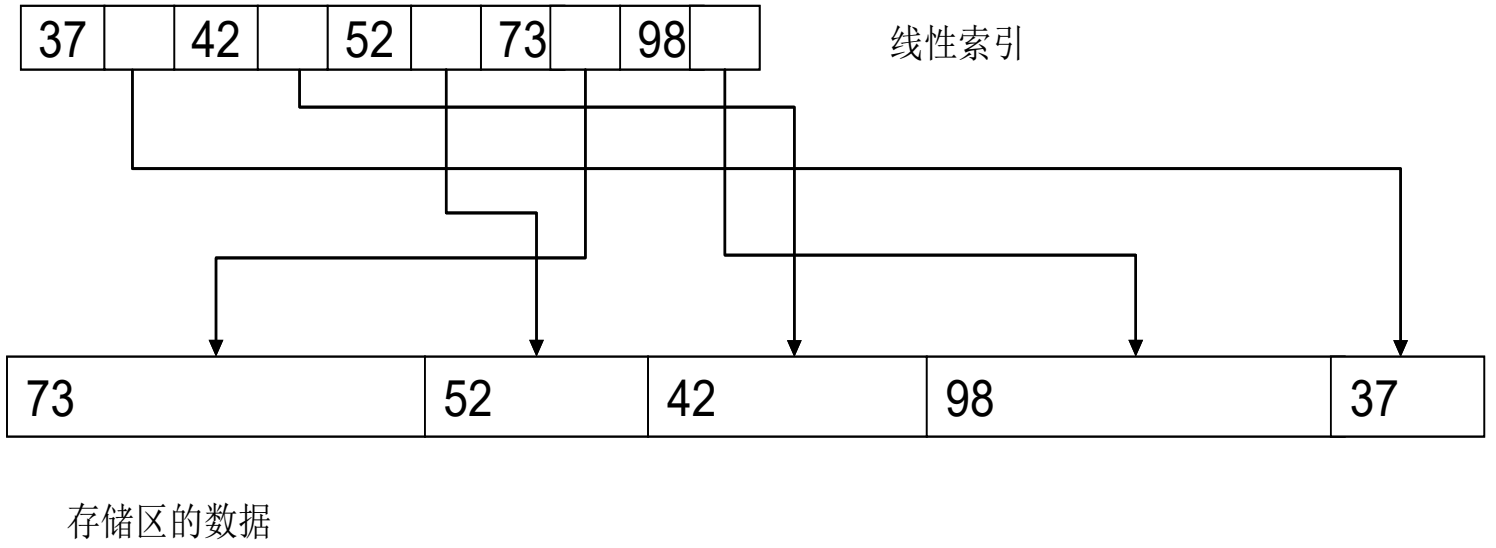
3、索引方法

目　录

索引方法(Con.)

- 是顺序存储法的一种推广
- 索引方法是要建造一个由整数域 Z 映射到存储地址域D 的函数Y：

  Z→D，把结点的整数索引值 z∈Z映射到结点的存储地址 d∈D 。 Y称为索引函数

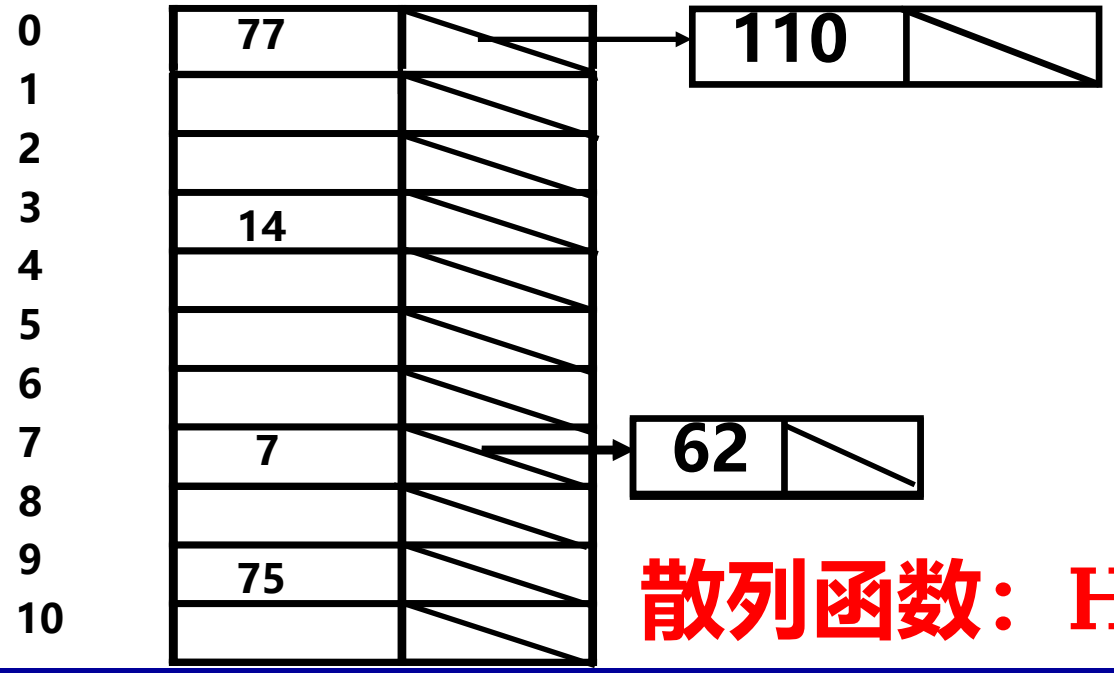


4、散列方法

- 散列是索引方法的扩展
- **散列函数**：将关键码 s **映射**到非负整数 z

$$h: S \rightarrow Z$$

对任意的 $s \in S$，散列函数 $h(s)=z$，$z \in Z$



散列函数：$H(k) = k \% 11$

小结

数据结构

- 逻辑结构
  - 节点
    - 基本数据类型

      整数、实数、布尔、字符，指针
    - 复合数据类型
  - 结构（关系）
    - 线性结构
    - 树型结构
    - 图型结构
- 存储结构
  - 顺序的方法
  - 链接的方法
  - 索引的方法
  - 散列的方法
- 数据运算

(一).抽象数据类型(ADT)

- 抽象问题得到解决，同类具体问题都可得到解决
- ADT是对多种可能的结构和实现的抽象
- 模块化思想的发展，提供了抽象数据类型的实现手段，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
- 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

抽象数据类型(续)

- 用数学方法定义对象集合和运算集合，仅通过运算的性质刻画数据对象，而独立于计算机中可能的表示方法。
- 目的：隐藏运算实现的细节和内部数据结构

抽象数据类型(续)

- 抽象数据类型由

  <数据对象，数据关系，数据操作>

  三个不可分割的部分组成的三元组：

- ADT抽象数据类型名 {

  数据对象D: <数据对象的定义>

  数据关系S: <数据关系的定义>

  数据操作P: <基本操作的定义>

  }ADT抽象数据类型名

ADT示例

```
template<class Type>
class className {
    private:                    //数据结构的取值类型与取值空间
        Type  dataList;         //私有变量，存储向量元素
        …
    public:                     //运算集
         methodName();          //定义对数据的操作
        …
};
```

(二). 算法

问题 —— 算法 —— 程序

目标：问题求解

- **问题（problem）—一个函数**
  - 从输入到输出的一种映射
- **算法（algorithm）—种方法**
  - 对特定问题求解过程的描述，是指令的有限序列
- **程序（program）**
  - 是算法在计算机程序设计语言中的实现

数据结构 + 算法 = ？

- Pascal之父、结构化程序设计先驱Niklaus Wirth最著名的一本书，叫作《数据结构 + 算法 = 程序设计》
- 程序设计的实质
  - 为计算机处理问题编制一组"指令“
- 需要解决两个 问题：算法和数据结构
  - 数据结构 = 问题的数学模型
  - 算法 = 处理问题的策略

“数据结构＋算法＝ 程序”

- **表达了数据结构与算法的内在联系，及其在程序中的地位**
  - 程序：在特定逻辑结构和存储表示的数据基础上，算法具体实现的描述
  - 数据结构与算法是程序设计中相辅相成、不可分割的两个方面

算法的性质

- **通用性**
  - 对参数化输入进行问题求解，保证计算结果的正确性
- **有效性**
  - 有限条指令序列能完成算法所要实现的功能
- **确定性**
  - 算法描述中的下一步应执行的步骤必须明确
- **有穷性**
  - 算法的执行必须在有限步内结束
  - 换句话说，算法不能含有死循环

算法分类

算法设计与分析是计算机科学的核心问题

- **穷举法(百钱买百鸡)——万能，低效**
  - 避免穷举测试
- **回溯(迷宫、八皇后)、搜索(DFS, BFS)**
  - 跳过无解分支、最优化问题的通法
- **贪心法**
  - **最优子结构——最优解**
  - **否则，只是快速得到次优解**

算法分类(续)

- **递归分治(二分检索、快速排序、分治排序)**
  - 子结构不重复
  - 分、治、合

- **动态规划(Floyd算法)**

——自底向上，利用中间结果，迅速构造

  - **最优子结构、重复子结构、无后效性**
  - **搜索中分支定界的特例**
  - **空间换时间**

算法分析

- 算法分析是对一个算法需要多少计算时间和存储空间的定量分析
- 判断所提出的算法是否符合现实情况
- 时间和空间复杂性
  - Tradeoff

Time is Important

算法复杂性度量

- 不能用绝对时间单位（如秒、分钟等）
  - 环境不同
    - 运行在巨型机上的算法会比PC机慢很多
  - 语言不同
    - C vs LISP
  - 可扩展性不同
    - 在不同输入规模上表现不同（100k vs 100M）

绝对运行时间的比较也是需要的

算法复杂性度量(续)

- 重要的不是具体的时间，而是算法复杂性与输入数据规模(N)之间的关系
  - 如对排序算法来说，输入规模一般就是待排序元素的个数
- 用“输入规模(N)“与所需“基本操作(B)”量级的函数关系来描述时间复杂性

示例

- 下面是一段对数组中的各个元素求和的代码：

```
for (i = sum = 0; i<n; i++)
    sum += a[i];
```

- **数据规模**： n
- **基本操作**：赋值运算
  - 在循环开始之前有2次赋值，分别对 i 和对sum进行；
  - 循环进行了n 次，每次循环中执行2次赋值，分别对 sum和对 i 进行更新操作，总计2n次；
  - 总共有f(n)=2 + 2n次赋值操作（资源开销）

分析框架：增长趋势

增长趋势

- 小规模输入在运行时间上的差别不足以将高效的算法和低效的算法区分开来

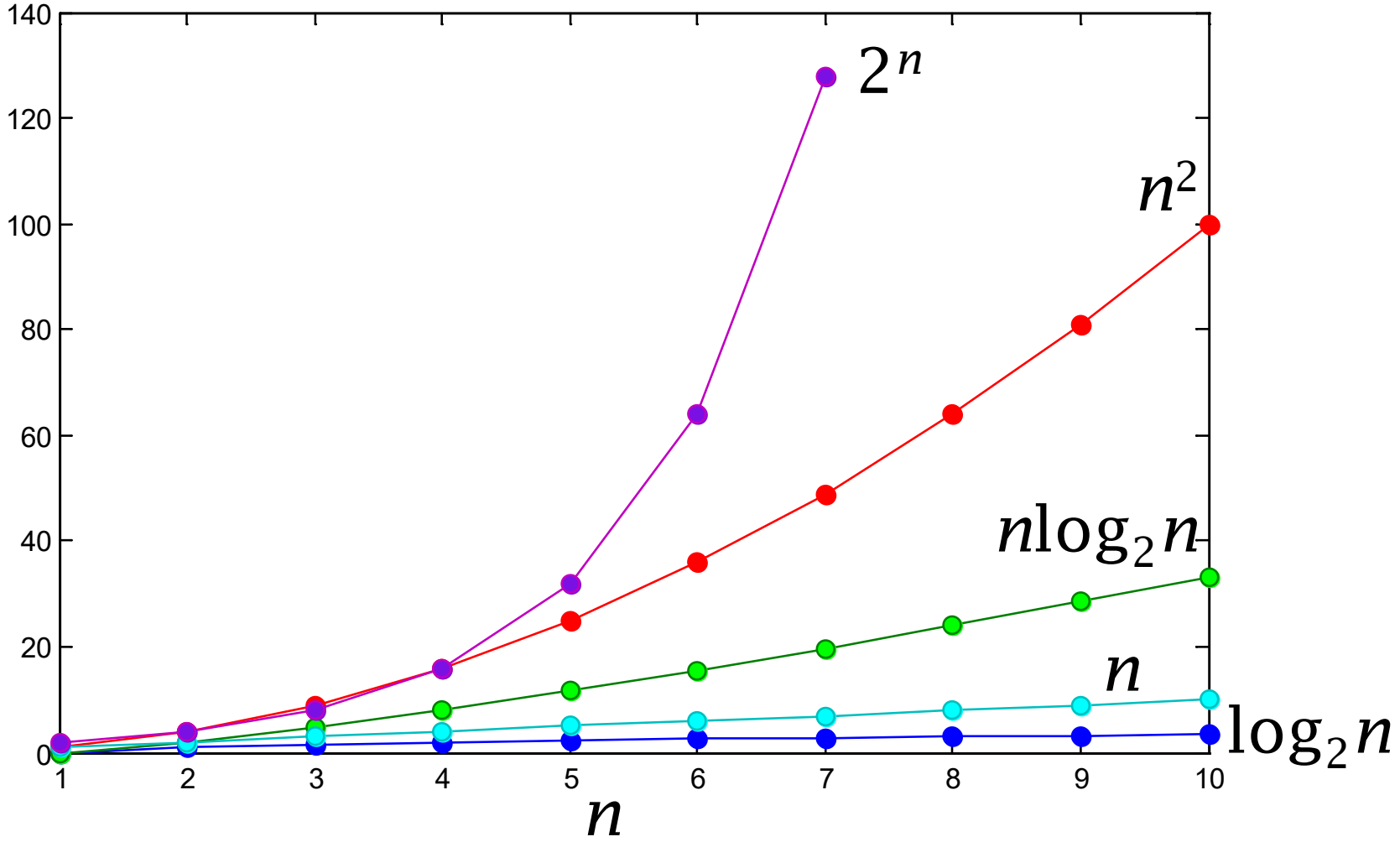
对于算法分析具有重要意义的函数值（有些是近似值）

n	$\log_2 n$	n	$n\log_2 n$	n^2	n^3	2^n	$n!$
10	3.3	10^1	$3.3\cdot10^1$	10^2	10^3	10^3	$3.6\cdot10^6$
10^2	6.6	10^2	$6.6\cdot10^2$	10^4	10^6	$1.3\cdot10^{30}$	$9.3\cdot10^{157}$
10^3	10	10^3	$1.0\cdot10^4$	10^6	10^9		
10^4	13	10^4	$1.3\cdot10^5$	10^8	10^{12}		
10^5	17	10^5	$1.7\cdot10^6$	10^{10}	10^{15}		
10^6	20	10^6	$2.0\cdot10^7$	10^{12}	10^{18}		

指数级复杂性算法只能用来解决小规模问题

常用函数的增长趋势

增长率函数曲线



渐进性分析方法

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

- **算法分析：估计当数据规模n逐步增大时，（时间或者空间）资源开销f(n)的增长趋势（函数形式）**
- **当n增大到一定值后，资源开销公式中影响最大的是n的幂次最高项，其他常数项和低幂次项可忽略不计**

1、大O表示法

- 定义1：如果存在正数c和n_0，使得对任意的$n \geq n_0$，都有

$$f(n) \leq cg(n)$$

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：函数g(n) 是函数 f(n) 取值的上限（upper bound）
- 大O表示提供了一种表达函数增长率上限的方法
- 一个函数增长率的上限可能不止一个。大O表示法给出了所有上限中最“紧”（也即最小）的那个上限

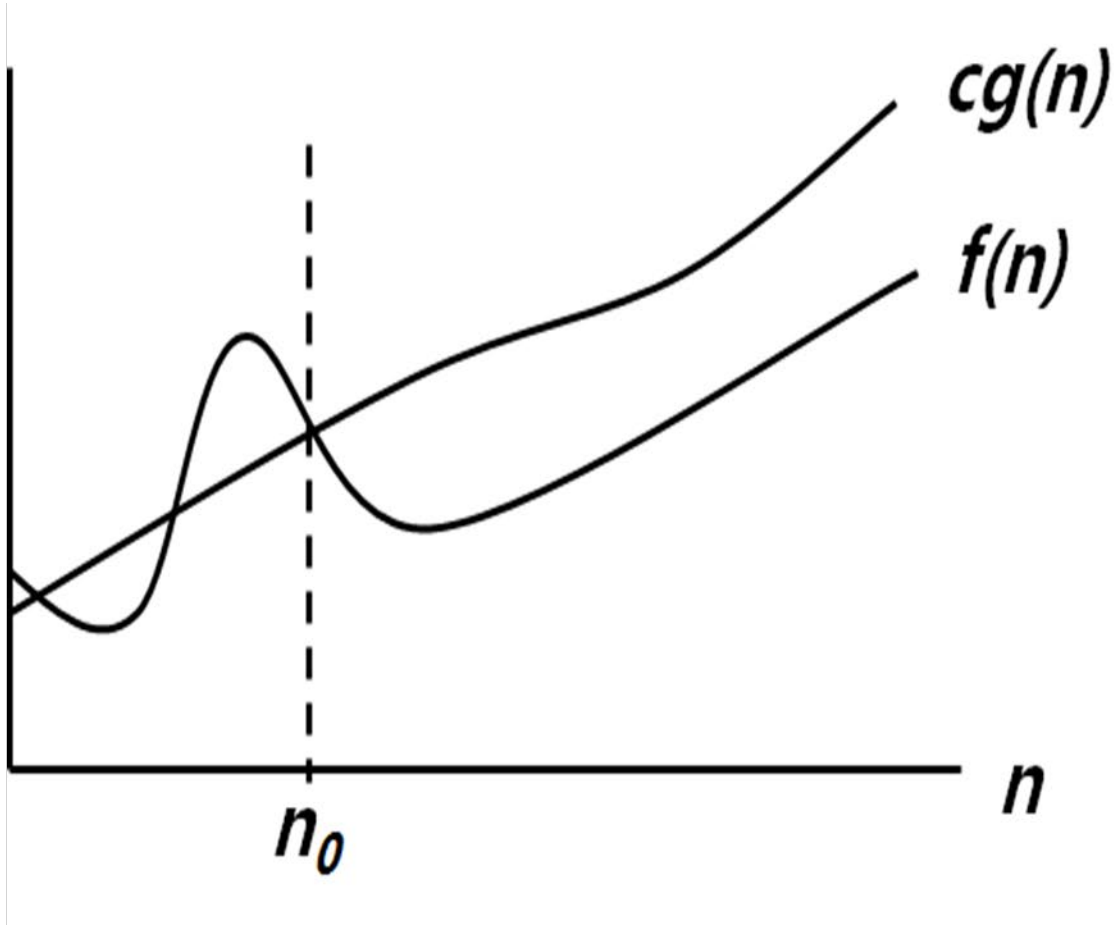
1、大O表示法

$3n \in O(n)$ $c=4$ $n0=1$

$n \in O(n^2)$

$100n+5 \in O(n^2)$

$n(n-1)/2 \in O(n^2)$



大O表示法的性质

- **若符号a是不依赖于n的任意常数**
  - 如果函数 f(n) 是O(g(n))的，g(n)是O(h(n))，那么f(n)是O(h(n))的
  - 如果函数f(n)是O(h(n))的，g(n)是O(h(n))，那么f(n) + g(n) 是O(h(n))的
  - 函数$a*n^k$是$O(n^k)$的，a不依赖于n
  - 若f(n) = cg(n)，则f(n) 是O(g(n))的
  - 对于任何正数a和b，且b≠1，函数$\log_a n$ 是$O(\log_b n)$的。即，**任何对数函数无论底数为何，都具有相同的增长率**
  - **对任何正数 a≠1，都有$\log_a n$是O(lg n)的，其中$\lg n = \log_2 n$**

大O表示法的运算规则

- **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**

2、Ω(Omega)表示法

- 定义2：如果存在正数c和n0，使得对所有的n≥n0，都有

$$f(n) \geq cg(n)$$

则称f(n)在集合Ω (g(n)) 中，或简称f(n) 是 Ω (g(n))的

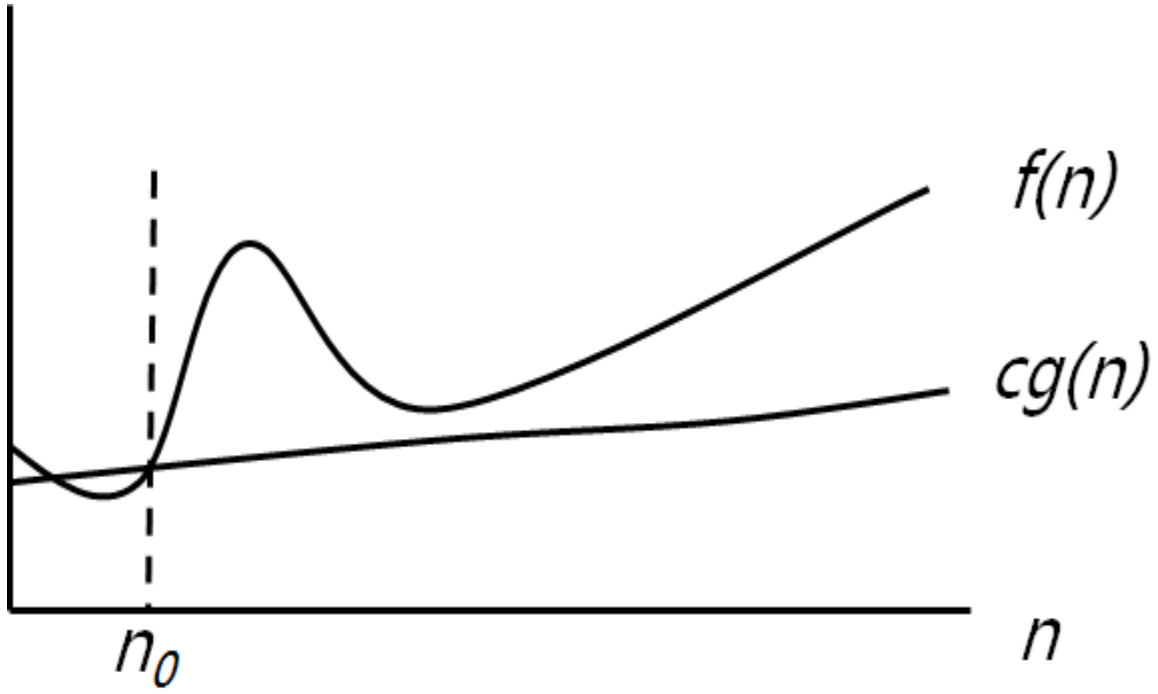
- 此定义说明了cg(n)是函数f(n)取值的下限（lower bound）
- Ω表示法是函数增值率所有下限中最“紧”（即最大）的下限

2、Ω(Omega)表示法

$n^3 \in \Omega(n^2)$

$n(n+1) \in \Omega(n^2)$

$4n^2+5 \in \Omega(n^2)$



3、Θ(Theta)表示法

- **当上、下界相同时则可用 Θ 表示法**
- **定义3：如果一个函数既在集合O (g(n))中又在集合Ω (g(n))中，则称其为Θ (g(n))。即存在正常数c_1, c_2，以及正整数n_0，使得对于任意的正整数n > n0，有下列两不等式同时成立：**

$$c_1\ g(n) \le f(n) \le c_2\ g(n)$$

- **例子：** $f(n) = 100 * n^2 + 5 * n + 500$
- **令** $g(n) = n^2$ **, 此时存在常数c1=100, c2=105, N=10, 当n>N时成立。**

$$c_1 g(n) \le f(n) \le c_2 g(n)$$

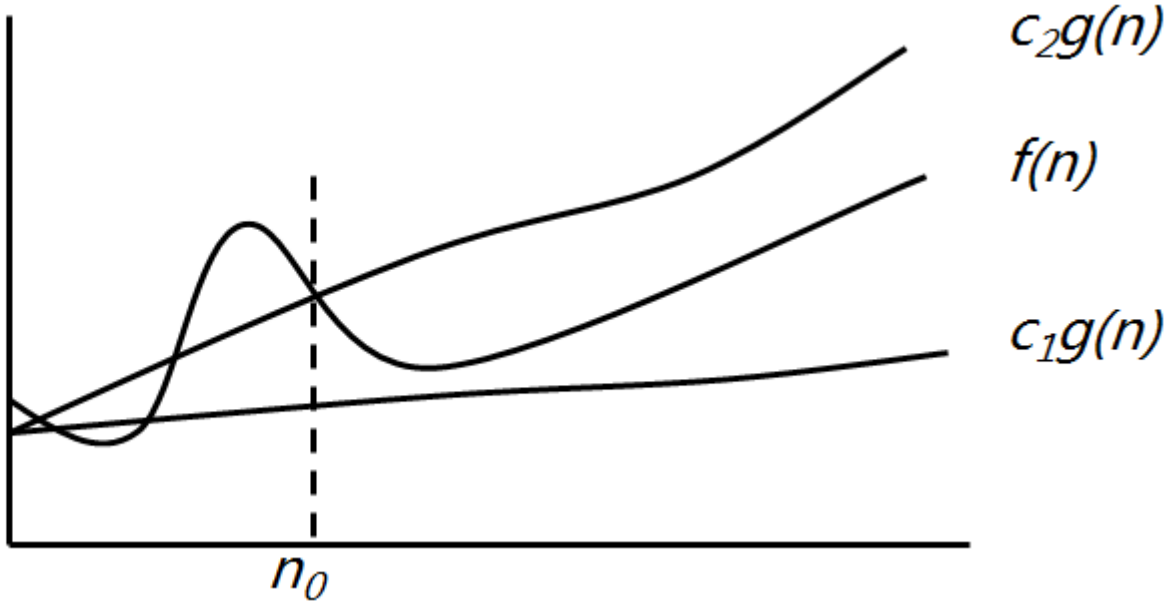
因此f(n)为 $\Theta(n^2)$

3、Θ(Theta)表示法

$n^2+3n+2 \in \Theta(n^2)$

$n(n-1)/2 \in \Theta(n^2)$

$4n^2+5 \in \Theta(n^2)$



算法分析示例（1）

示例1：依次求出给定数组的所有子数组中各元素之和：

```
for  (i = 0; i < n; i++)
        for  (j = 1, sum = a[0];  j <= i;  j++)
                sum += a[j];
```

- 循环开始前，有1次对 i 的赋值操作
- 外层循环共进行n次，每个循环中包含一个内层循环，以及对i，j，sum进行3次赋值操作
  - 每次内层循环执行2次赋值操作（更新sum 和 j），执行i次（i=1,2, …,n-1）
- 整个程序总共执行的赋值操作为

$$1+3n+\sum_{i=1}^{n-1}2i=1+3n+2(1+2+....n-1)=1+3n+n(n-1)=O(n)+O(n^2)=O(n^2)$$

算法分析示例（2）

示例2：若只对每个子数组的后5个元素求和，则相应的代码可采用下面的方式：

```
for ( i=4;  i<n;  i++)
        for (j = i-3, sum = a[i-4]; j <= i; j++)
                sum += a[j];
```

- 外层循环进行n-4次
- 对每个 i 而言，内层循环执行4次，每次操作次数和 i 的大小无关：11次赋值操作
- 整个代码总共进行 1 + 11*(n-4) = O(n)次
- 看似双重循环，其实线性时间

例题

下面函数的时间复杂度是

```
int work(int n, int m) {
    int sum = 0;
    for (int i = 1; i <= m; i *= 2)
        for (int j = 1; j <= n; j *= 2) {
            sum += i * j;
        }
    return sum;
}
```

O(log n * log m)

最坏、最好、和平均情况

- 算法分析受条件分支的影响
  - 算法的增长率估计往往由于算法中的条件分支而遇到困难
  - 分支走向又受输入数据取值的影响，因此很多算法都无法得出独立于输入数据的渐进估计
- 估计方法
  - 最坏情况估计
  - 最好情况估计
  - 平均情况估计

平均情况下的复杂性分析

- 平均情况下的算法复杂度分析，要考虑算法的所有的输入情况，确定每种情况下的输入数目
- 简单情况下，需要考虑每种输入情况的概率

$$C_{avg} = \sum_i p(input_i)steps(input_i)$$

$p(input_i)$为第i 种输入的出现概率

$steps(input_i)$为算法处理第i种输入时所需的基本操作数目

复杂性分析(Con.)

- 对于时间开销，不关注最好估计，而关注最坏估计
  - 特别是处理应急事件，计算机系统必须在规定的响应时间内做完紧急事件处理
- 对于多数算法而言，最坏情况和平均情况的估计，其时间开销公式虽然不同，但往往只是常数因子大小的区别，或者常数项的大小区别。因此不会影响渐进分析的增长率函数估计

时间和空间的折衷

- 空间开销也可以实行类似的渐进分析方法
  - 指除数据本身之外的存储开销
- 静态存储结构
  - 空间开销估计往往容易
- 动态存储结构
  - 算法运行过程中会有数量级地增大或缩小
  - 这种情况的空间开销的分析和估计是十分必要的

时空折衷（Time and Space Trade-off ）

- **空间换时间**：为改善算法的时间开销，可通过增大空间开销而设计出时间开销小的算法
- **时间换空间**：为缩小算法的空间开销，通过增大时间开销来换取存储空间的节省

总结

- 学完了本章，你达到下述要求
  - 数据结构的定义
  - 抽象数据类型
  - 算法的特性及分类
  - 算法的效率度量

再见…

联系信息：

电子邮件：gjsong@pku.edu.cn

电　　话：62754785

办公地点：理科2号楼2307室