数据结构

袁平波　2013.1

第一章 绪论

1.1《数据结构》讨论范畴

1.2 基本概念和术语

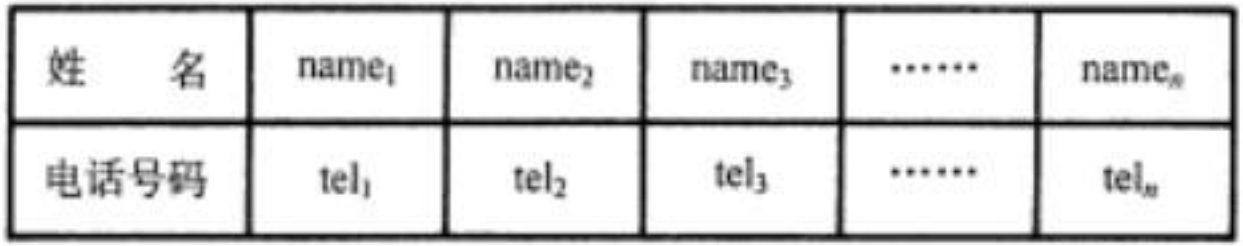
1.3 抽象数据类型的表示与实现

1.4 算法和算法分析

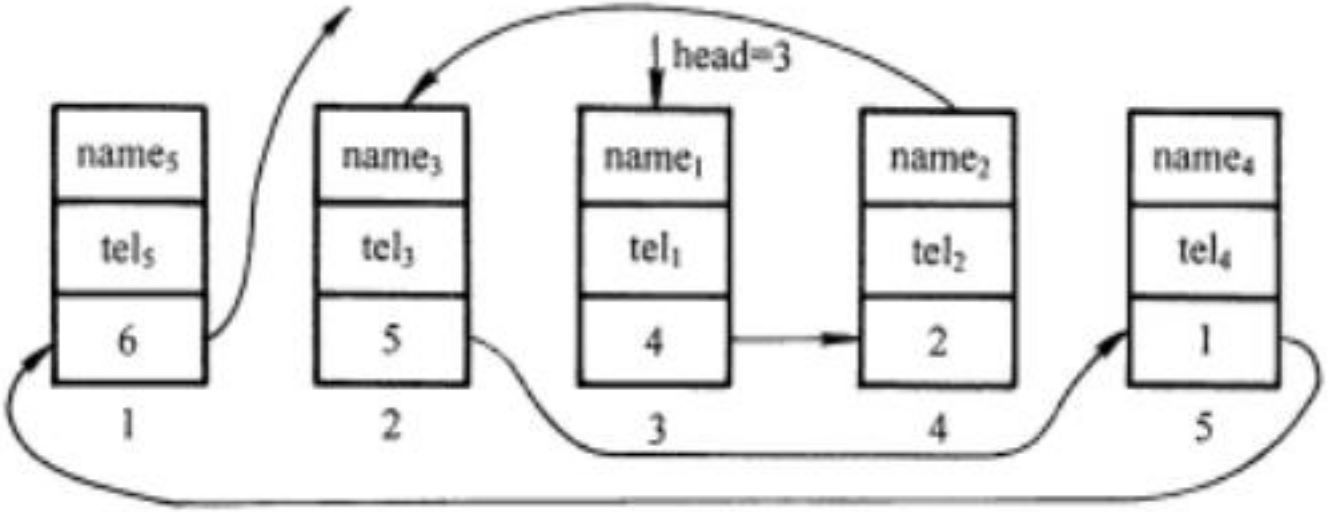
1.1《数据结构》研究什么

- ☞算法+数据结构=程序设计
- ☞ 数据结构是讨论非数值类问题的对象描述、信息组织方法及其相应的操作

[例]、设有一个电话号码薄，有N个人的姓名和电话号 码。要求设计一个程序，按人名查找号码，若不存在则给出不存在的信息。



(a) 顺序存储



(b) 链式存储

1.2 基本概念和术语

- **数据：**数据是客观事物的符号表示。
- **数据元素：**数据的基本单位，通常看作一个整体。
- **数据对象：**性质相同的数据元素的集合。
- **关系：**集合中元素之间的相关性。
- **数据结构：**特性相同的数据元素构成的集合中，如果在数据元素之间存在一种或多种特定的关系，则称之为数据结构 。

Data-Structure=(D,S)

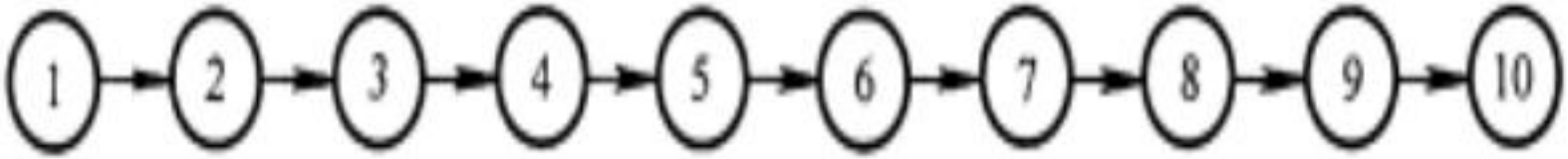
D是元素有限集，S是关系有限集。

四类基本结构：1)集合 2)线性 3)树形 4)网状

[例] linear=(D,R)

D={1,2,3,4,5,6,7,8,9,10}

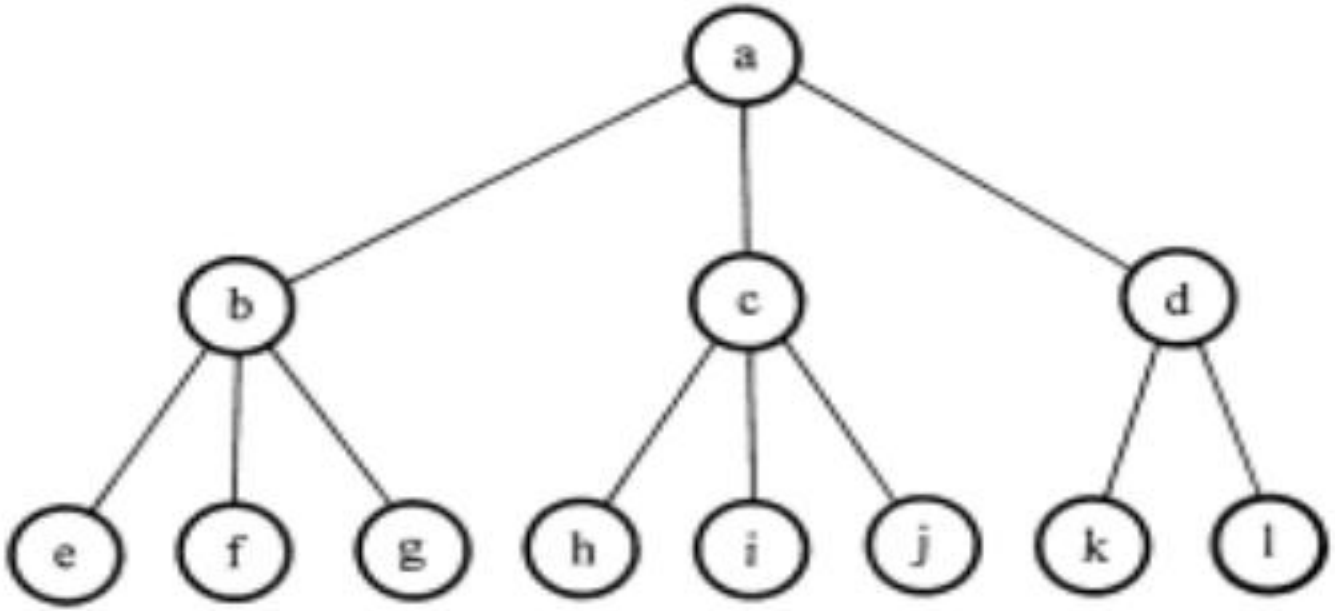
R={<1,2>,<2,3>,<3,4>,<4,5>,<5,6>,
<6,7>,<7,8>,<8,9>,<9,10>}



[例]tree=（D,R）

D={a, b, c, d, e, f, g, h, i, j, k, l}

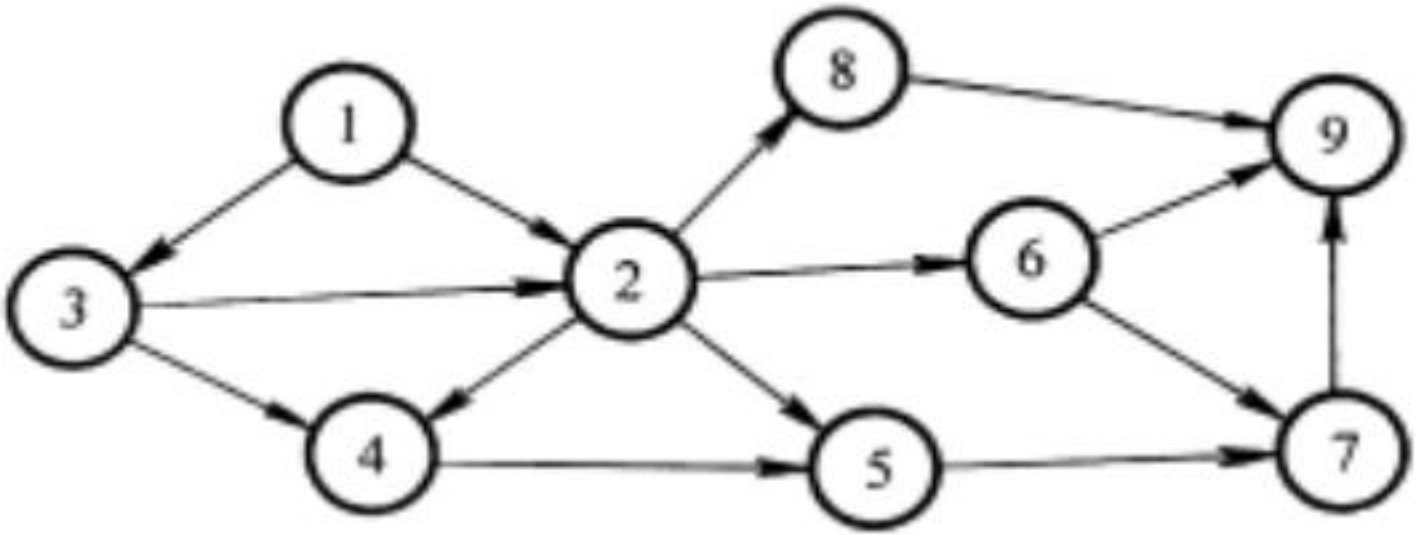
R={<a,b>,<a,c>,<a,d>,<b,e>,<b,f>,<b,g>,<c,h>,<c,i>,
<c,i>, <d,k>,<d,l>}



[例] graph=(D,R)

D={1, 2, 3, 4, 5, 6, 7, 8, 9}

R={<1,2>,<1,3>,<2,4>,<2,5>,<2,6>,<2,8>,<3,2>,<3,4>, <4,5>,<5,7>,<6,7>,<6,9>,<7,9>,<8,9>}



其他概念与术语

- 逻辑结构/物理结构(存储结构)
- 位(bit)/字节(byte)
- 元素(Element)/结点(Node)/数据域(DataField)
- 顺序存储/链式存储
- **数据类型**：值的集合和定义在这个值集上的一组操作的总称。分**原子**和**结构**两种类型。
- **抽象数据类型**(Abstract Data Type)：
  一个数学模型以及定义在该模型上的一组操作。

1.3 抽象数据类型的表示与实现

ADT=（D，S，P）

D是数据对象，S是D上的关系的集合, P是对D的基本操作的集合。

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

基本操作的 定义格式：

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〉

[例]抽象数据类型三元组的定义

ADT Triplet{

数据对象:D={e1,e2,e3|e1,e2,e3∈Elemset}

数据关系:R={<e1,e2>,<e2,e3>}

基本操作:

InitTriplet(&T,v1,v2,v3)

操作结果:构造三元组T，元素e1,e2,e3分别赋予v1,v2,v3

Get(T,i,&e)

初始条件:三元组T存在，1≤i≤3。

操作结果:用e返回T的第i元的值。

Max(T,&e)

初始条件:三元组T存在。

操作结果:用e返回T中三个元素的最大值。

… …

}ADT Triplet

操作中的参数有两种

1）赋值参数

```
void add (int a, int b, int * c){
      *c=a +b;}
add(2,3,&c);
```

2）引用参数

```
void add (int a, int b, int &c){
      c=a +b        }
add(2,3,c);
```

类C语言——介于伪码和C语言之间

1）预定义常量和类型

```
#define TRUE 1
#typedef int Status
```

2）数据元素类型约定为ElemType，由用户使用时自行定义。

3）基本操作的算法用函数描述：

```
函数类型  函数名（函数参数表）{
        //算法说明
        语句序列
}//函数名
```

4）赋值语句

简单赋值：变量名=表达式；

串联赋值：变量名1=变量名2=…=表达式；

成组赋值：变量名1[起始下标..终止下标]=变量名2[起始下标..终止下标]；

交换赋值：变量名1←→变量名2；

条件赋值：变量名=条件表达式?表达式T：表达式F

5)选择语句

条件语句1：if(表达式)语句;

条件语句2：if(表达式)语句; else 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;
}
```

6)循环语句

for语句:　　for(赋初值;条件;修改表达式)语句;

while语句:　　while(条件)语句;

do-while语句:　do{语句序列}while(条件);

7)结束语句

函数结束语句　return; return 表达式;

异常结束语句　exit;

8)输入输出语句

输入语句　scanf

输出语句　printf

9)注释　　单行注释用“//”

10)基本函数: max,min,abs,floor,ceil,eof,eoln等。

11)逻辑运算

与运算&&　　或运算||

1.4算法和算法分析

1、**算法**:对问题求解步骤的描述，是一个确定的、有限长的操作序列。

1)有穷性 2)确定性 3)可行性 4)输入 5)输出

2、算法设计要求

1)正确性 2)可读性 3)健壮性 4)效率与低存储需求

3、算法的描述:类C语言

4、算法效率衡量方法与准则 ：

时间复杂度：T（n）=0（f(n)）

空间复杂度：S（n）=0（g(n)）

“O” 的形式定义

若f(n)是正整数n的一个函数，则x_n＝O(f(n))表示存在一个正的常数M,m，使得当n≥n0时都满足

$m|f(n)| \leqslant |x_n| \leqslant M|f(n)|$;

换句话就是说这当整型自变量n趋向于无穷大时，两者的比值是一个不等于0的常数。

$$\lim_{n\to\infty}\frac{|X_n|}{|f(n)|}=C$$

分析下列三个程序段：

①、x++;s=0; O（1）

②、for (i=1;i<=n;i++) {x++;s+=x;} O（n）

③、for(i=1;i<=n;i++)

for (j=1;j<=n; j++) {x++; s+=x;} O（n^2）

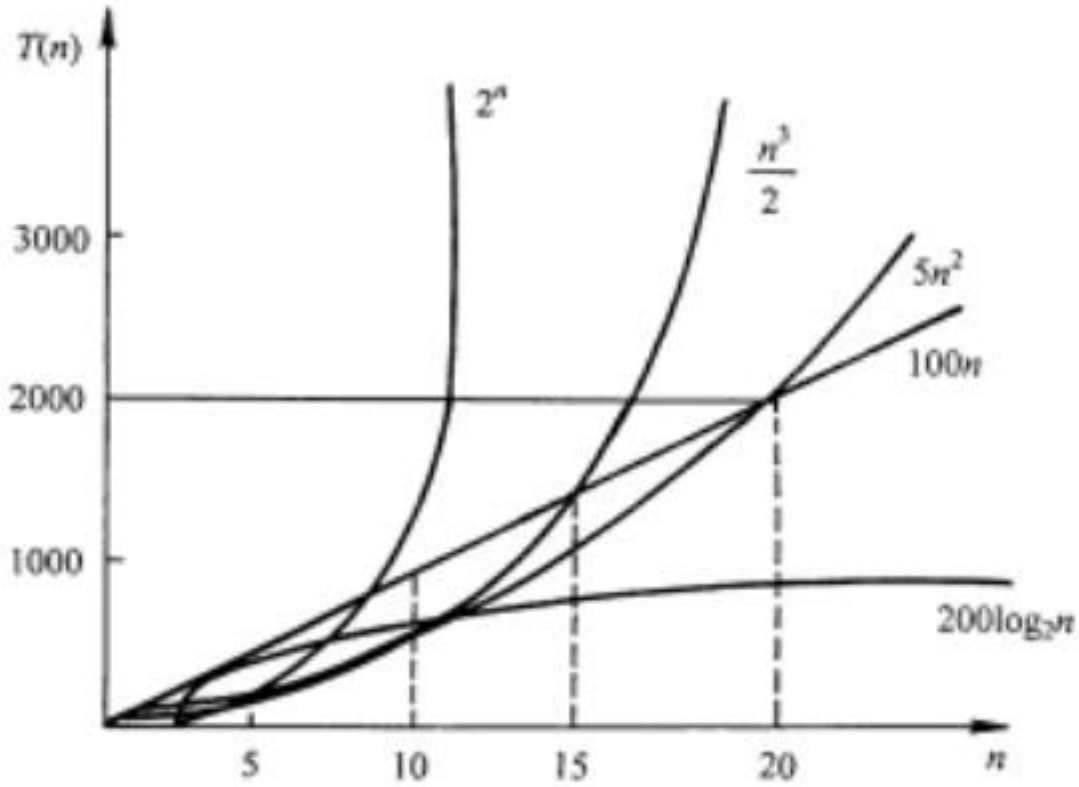
④、$f(n)=2n^3+5n^2+7n+c$;（c为某一常数），则

T（n）= O (f(n))=O（n^3）。

[例] 起泡排序　　算法 1.3

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

时间复杂度$O(n^2)$



常见函数的增长率