数 据 结 构

周洪涛
王少波

2024年02月

周洪涛

- Wx/qq/mobile phone:15953276901
- zhouht@cupk.edu.cn
- 周洪涛简历：中国石油大学(北京)克拉玛依校区石油学院 (cupk.edu.cn)

课堂讨论

前期学习：C语言，已经学到了哪些？

如何学好：预习、课堂讨论、课后；读3遍书

上课：一起学习，一起读书

概念、知识，操作与技能

前　言

- 数据结构是系统软件和应用软件研制者的必修课程。
- 数据结构主要解决非数值计算应用问题。
- 每种数据结构类型描述层次——
  概念层、逻辑层、存储层、运算实现层 。
- 数据结构描述的内容看上去如同程序，
  但不是程序，它是程序设计思想的抽象化 。
- 数据结构既是一门理论性很强的课程，同时又是一门实践性很强的课程。

第1章 绪 论

- **信息表示：信息是由信息元素的值及信息元素之间的相互关系（逻辑顺序）和信息元素在计算机中的存储方式（物理顺序）共同组成。**
- **计算机领域中“计算”有别于数学概念中的一般计算，通常将“计算”又称为“运算”或“操作”。**

运算或操作的内涵：

公式化运算：四则运算和各种函数运算

非公式化运算：数据存取、插入、删除、查找、排序和遍历等运算。

计算机应用主要包括两个方面：

数值计算和非数值计算。

数值计算问题：程序设计主要围绕程序设计技巧，是典型地以程序为中心的设计过程。

- 数值计算特点：是计算过程复杂，数据类型相对简单、数据量相对较少；

非数值计算问题：是以复杂的数据为中心，研究数据的合理组织形式，并设计出基于合理数据组织结构下的高效程序

- 非数值计算特点：计算过程相对简单，数据类型相对复杂。

非数值计算中数据的组织排列从某种意义上决定着非数值计算应用有效性。

- 。

1.1 什么是数据结构

1.1.1 数据结构相关事例

问题一：电话号码簿的使用及字典的使用。

构建电话号码簿的多级分类目录（数据组织结构）

大、小类别可以看作电话号码簿的目录或索引。

缺少类别的电话号码簿无人使用（数据的大量性）除非少量电话号码。

从特定的大类别中查找小类别直至所需要的电话号码（计算方法）。

事例：

电话号码簿

行业名称	页码
党政机关	7
大学	12
企业	25
旅游	32

1

行业名称	页码
省委	55
市委	127
区委	224

7

行业名称	页码
综合大学	325
理工类大学	327
人文类大学	334

12

单位名称	电话
省委办公厅一处	88060001
省委办公厅二处	88060002

55

单位名称	电话
市委办公一处	85800203
市委办公二处	85800105

127

单位名称	电话
华中科技大学	87870001
武汉大学	86880206

325

图1.1. 电话号码簿

问题二：车皮调度问题

- 有若干个发往同一方向不同城市的货车车皮以随机的次序到达货站的进车道，货站由三部分构成：入轨道，出轨道和调度道（缓冲铁轨）。
- 货车发出本站时，列车的尾部所挂接的车皮应该是发往距本货站最近的车皮，这样可以保证到达某车站时从尾部“甩下”若干到站的车皮。
- 车皮发出本站前应调整它们的次序：调整为小编号在前，大编号在后的排列次序
- 实现调整过程，货运站通过调度车道（缓冲铁轨）完成。

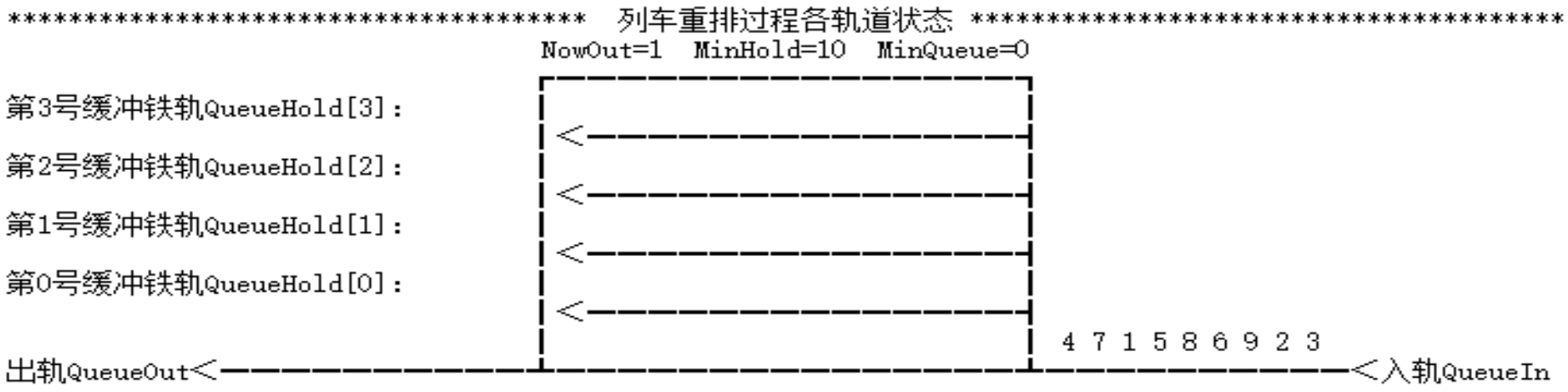


图1.2 车厢调度转轨（初态）

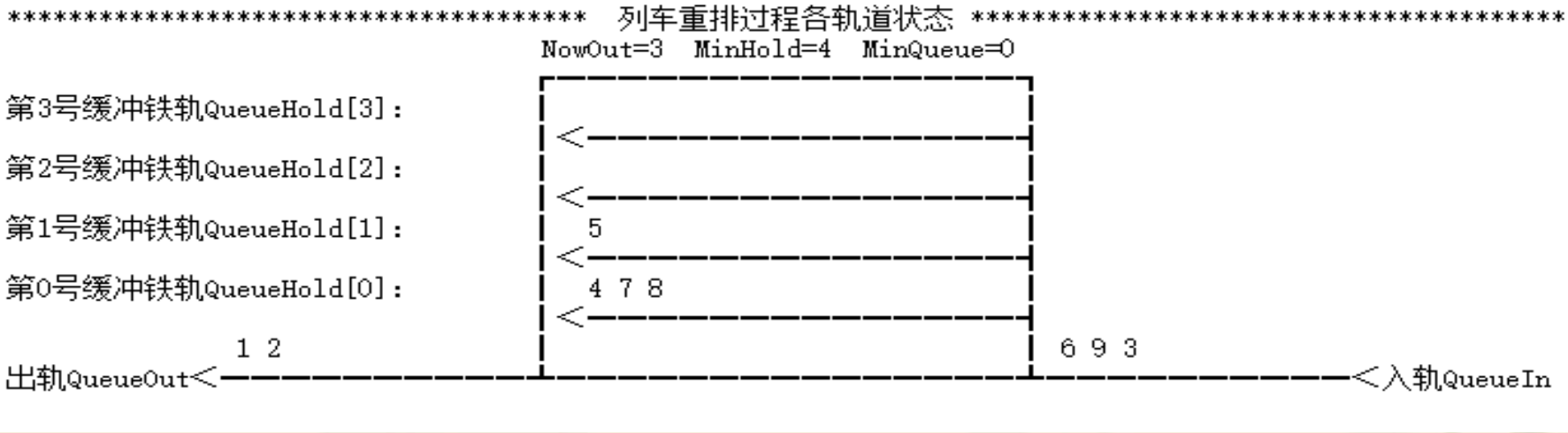


图1.3 车厢调度转轨

问题三：城市之间要架设电话通信线路。图1.1.3和图1.1.4

- 电话通信线路要保证各城市间互通，但不一定两两城市间都要连线
- 又要使架设成本最低：使用的线路，人工成本等。
  就是数据结构中图结构的应用--**最小生成树**。
- 实现过程，构造最小生成树。

架设电话通信线路

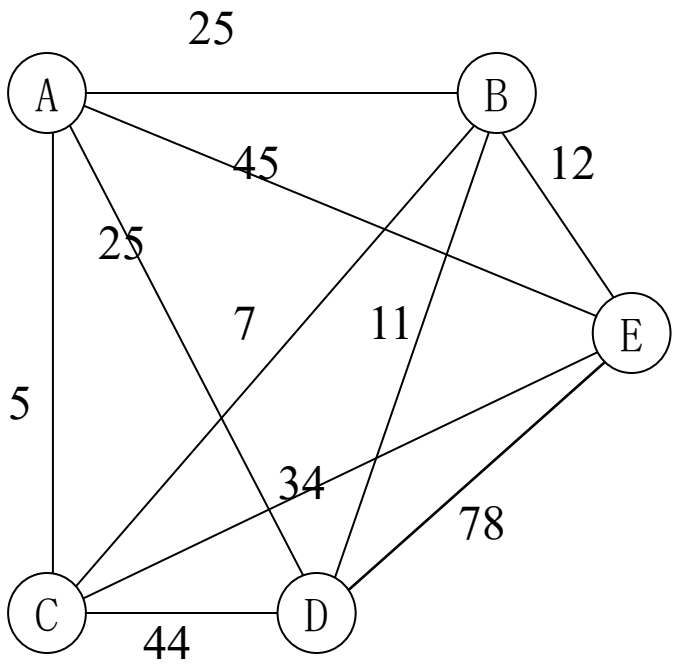


图1.4 城市连接及距离

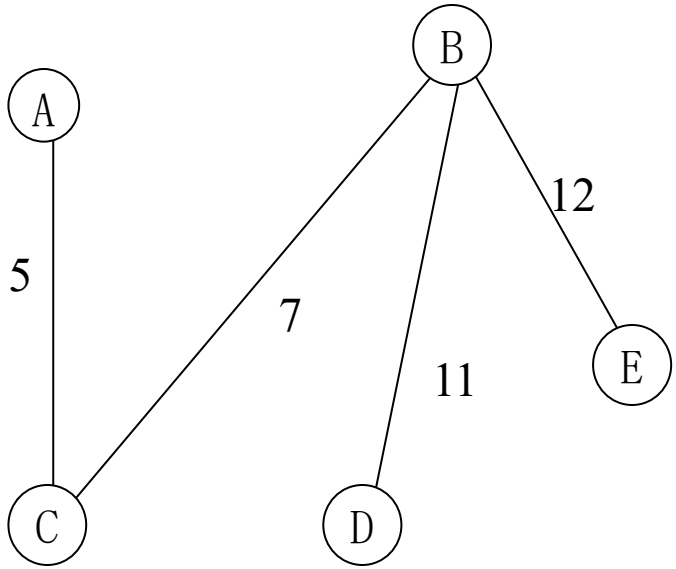


图1.5 最小生成树

1.1.2 数据结构的定义

定义：数据结构就是研究计算机中大量数据存储的组织形式，并定义且实现对数据相应的高效运算，以提高计算机的数据处理能力的一门科学。

定义全释：

计算机中大量数据问题；

数据存储的组织结构；

定义且实现相应高效的运算。

数据的组织形式（结构） 是基础

“运算”或 “操作”是数据结构讨论内容的一个核心

数据的**组织形式（结构）**具有两个层面：

- **逻辑结构**：是与计算机本身无关的“逻辑组织结构”它的构成是由数据的值、数据与数据之间的关联方式两个部分组成。
- **物理结构**：是将具有逻辑组织结构的数据在计算机的存储介质上如何存放的“物理组织结构”。

“**运算**”或“**操作**”是数据结构讨论内容的一个核心问题。就是“**算法设计**”。

- 算法不仅要实现问题的要求，而且，应该是**高效**地完成。低效的算法无法满足用户的需求或根本不能运用于实际。低效的处理算法设计的程序即使运用高速运算的计算机也可能不能满足用户的处理要求。

1.2 数据结构的相关概念

1.2.1 数据和信息

数据：数据是信息以某一类特定符号表示的形式，是计算机程序加工的对象。

信息：是描述现实中客观事物的数据集合。

1.2.2 数据元素（图1.2.1）

- **数据元素**：是数据集合中的个体，是数据组成的基本单位。
- 数据通常是由若干个数据元素组成，数据元素是**不可再分**的最小单位。构成数据元素的每个项目称为“**数据项**”。

如：商品数据元素是由：【商品名、商品编号、商品价格、商品数量】四个数据项构成

- 有的数据项是由单一的数据类型组成，称为**原子项**。如【商品名】的类型为字符串。
- 有的数据项又可再分为若干个子数据项组成，称为**组合项**。如【商品价格】是由【出厂价格、批发价格、零售价格】三个实数类型的数据项构成。

数据元素

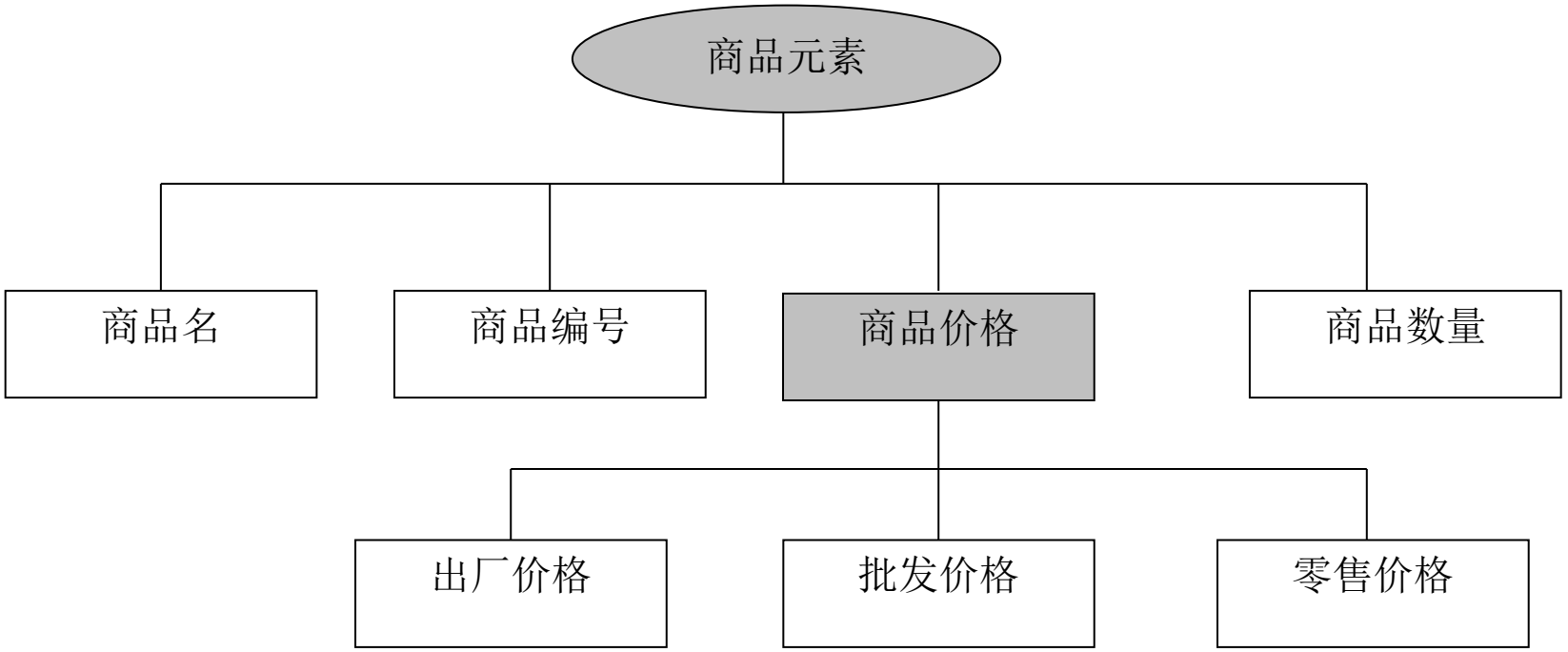


图1.6 商品数据元素

数据元素在数据结构中通常又可以表述为：
“记录”或“结点”或“顶点”。

- 在计算机的高级语言中又可以有不同的表述：
  C或C++语言：定义为“结构类型”（STRUCTURE TYPE）
  PASCAL语言：定义为“记录类型”（RECORD TYPE）
  数据库：定义为“元组”（TUPLE）

- 关键字或关键码（key）：在数据元素的多个数据项中能起标识作用的数据项。
  主关键码：能唯一标识一个数据元素的关键码。
  次关键码或次码：不能唯一标识一个数据元素的关键码。

1.2.3 结构类型

数据结构中结构类型分为:

逻辑结构和物理结构

1. 逻辑结构

逻辑结构描述数据元素与数据元素之间的关联方式，简称为关系，表示的是事物本身的内在联系。

如：前后关系、上下级关系、父子关系等。

逻辑结构可分为两类:

线性结构和非线性结构。

- 在数据结构中线性结构分为：
  线性表、堆栈、队列等
- 线性结构的特点：

数据元素 {…,a,b,c,…} 之间存在前后次序，排在某个数据元素b前面的数据元素a称为b的直接前驱元素，而数据元素b则称为数据元素a的直接后继元素，对于某个数据元素，如果存在直接前驱元素或直接后继元素，则都是唯一的。

线性结构中数据元素之间的正逆关系都是“一对一”的。

{ {彭亮，97} ， {王明，95} ， {李智，90} ， {刘丹，88} ， {肖象，78} }

图1.7线性结构的顺序映象

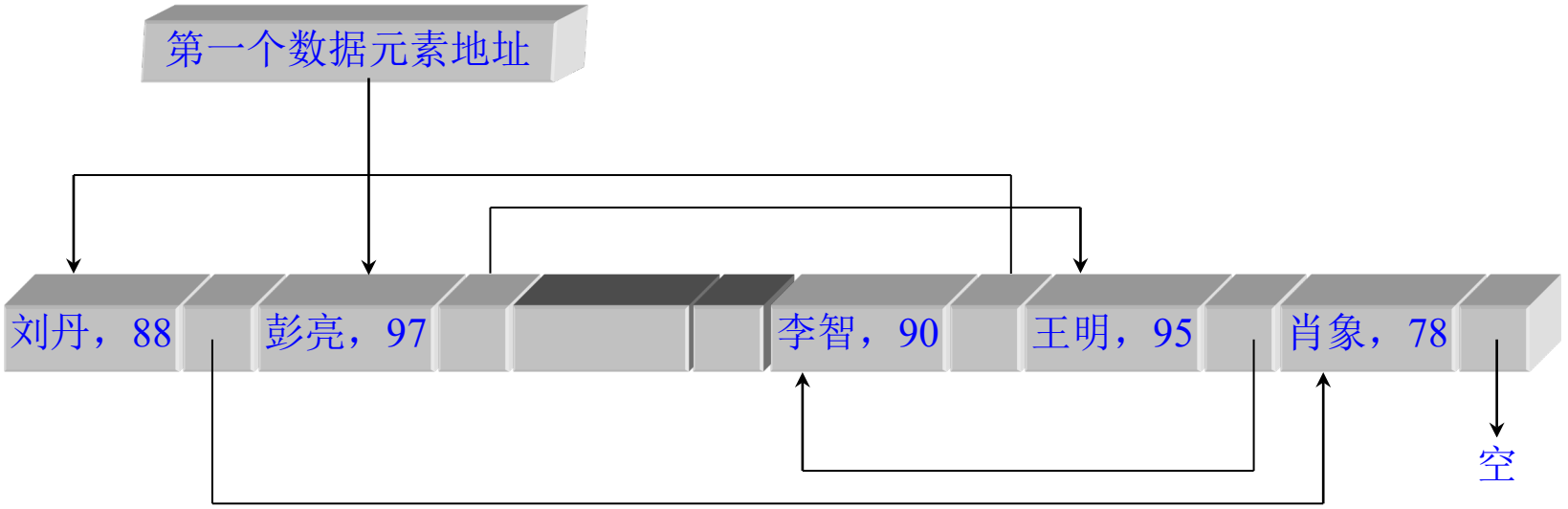


图1.10　线性结构的链式映象

- 在数据结构中非线性结构分为：图1.10 和图1.12 树形结构、图状或网状结构、纯集合结构。
- 非线性结构的特点：

数据元素不一定存在确定的前后次序，甚至是无序的，数据元素之间存在从属、或互为从属、或离散关系。

① 树型结构中，数据元素之间存在着“一对多”的从属关系。

② 图或网状结构中，数据元素之间存在着“多对多”的互为从属关系。

③ 在纯集合结构中，数据元素具有“同属于一个集合”的关系。

所有逻辑结构类型加以限制条件，都可以看作集合结构类型或纯集合结构类型的特例。

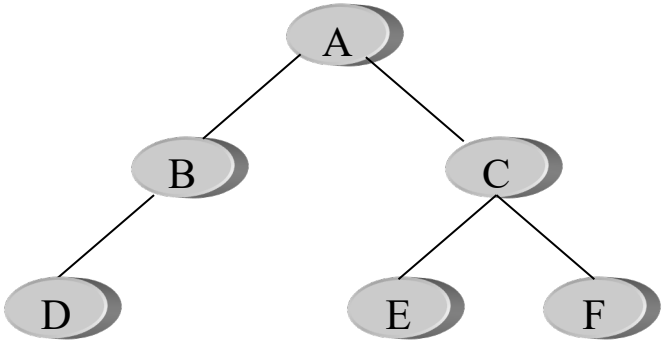


图1.11 二分枝的树逻辑结构

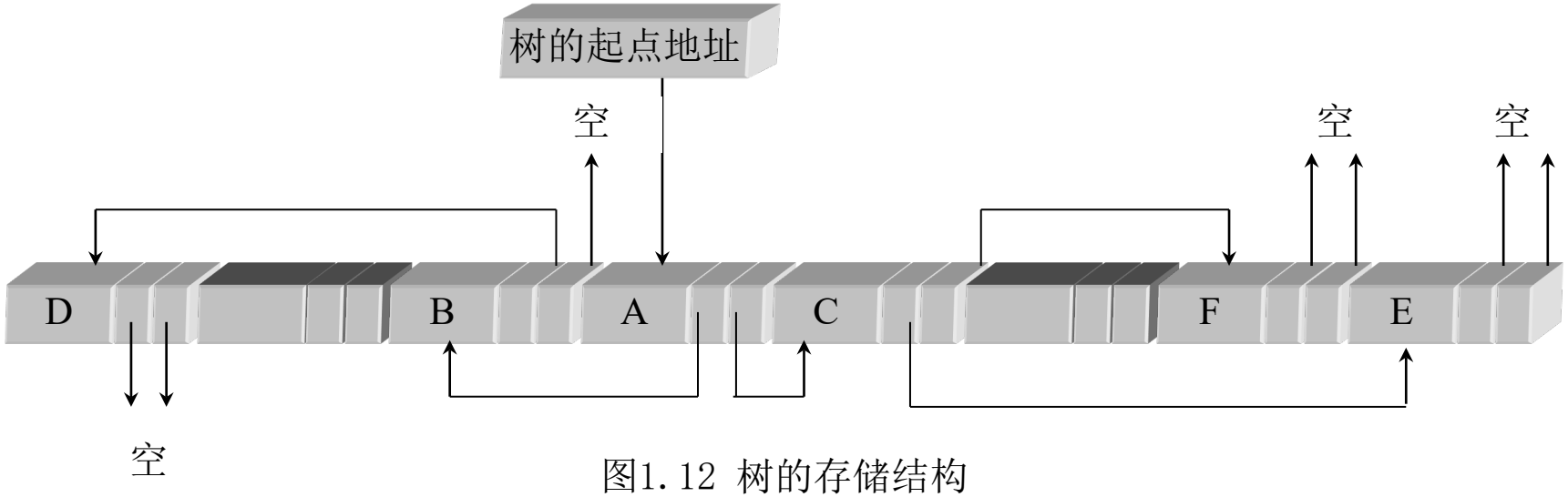


图1.12 树的存储结构

2.物理结构

物理结构:也称为存储结构，是逻辑结构的数据元素在计算机的物理存储空间上地映象（存储），映象不仅包含数据元素本身，而且包含着数据元素之间的关联方式，即关系的映象。

映象可以分为：

顺序映象和非顺序映象。

也可以描述为顺序存储和非顺序存储。

1）顺序映象

顺序映象:是指数据元素在一块**连续地**物理存储空间上存储，物理存储空间只用于存放数据元素**值**本身。

数据元素之间的**关联**以两个数据元素存储的**相邻关系**来表示，

或**相对位置关系**来表示

或通过某个**函数**来表示。

或者说，利用数据元素在存储空间上的**相对位置**来表示数据元素之间的**逻辑关系**。

逻辑结构和物理结构

{ {彭亮，97}，{王明，95}，{李智，90}，{刘丹，88}，{肖象，78} }

图1.7 成绩信息的顺序映象

	1	2	3	4	5
1	0	0	0	0	0
2	A	0	0	0	0
3	B	C	0	0	0
4	D	E	F	0	0
5	G	H	I	J	0

存储映像函数：

$$f(i,j)=((i-1)*(i-2))/2+j$$

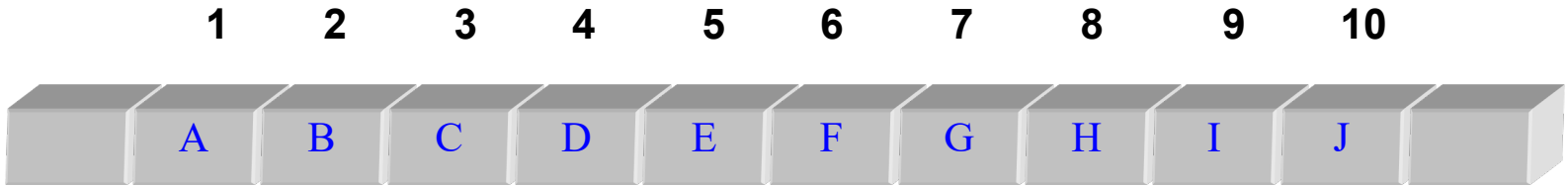


图1.9 下三角矩阵的顺序映象

2)非顺序映象

非顺序映象：是指数据元素在物理存储空间上非连续地存储，物理存储空间不仅存放数据元素本身，而且为实现数据元素之间的关联（逻辑结构），在每个数据元素存储的**相邻空间**中存储该数据元素关联的另一个或多个数据元素的**起始地址**。

也可以说数据元素之间的关系是由附加“**链接或指针**”来表示。**图1.11 和图1.12**

非顺序映象存储结构中，数据元素的逻辑结构一般在物理空间上不是以物理空间的相邻来表现，而是在每个数据元素值所占用空间的相邻空间中存放该数据元素所关联的另一个或多个数据元素的位置，通常将这个空间称为**链地址空间**。

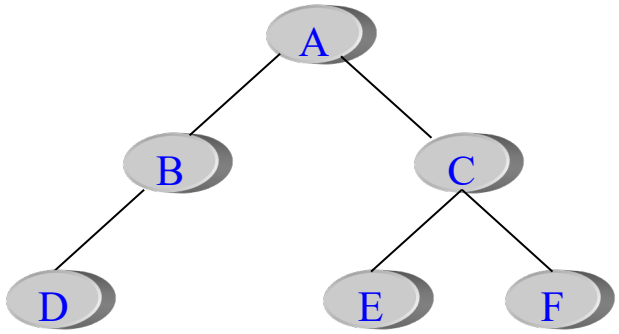


图1.11　二分枝的树逻辑结构

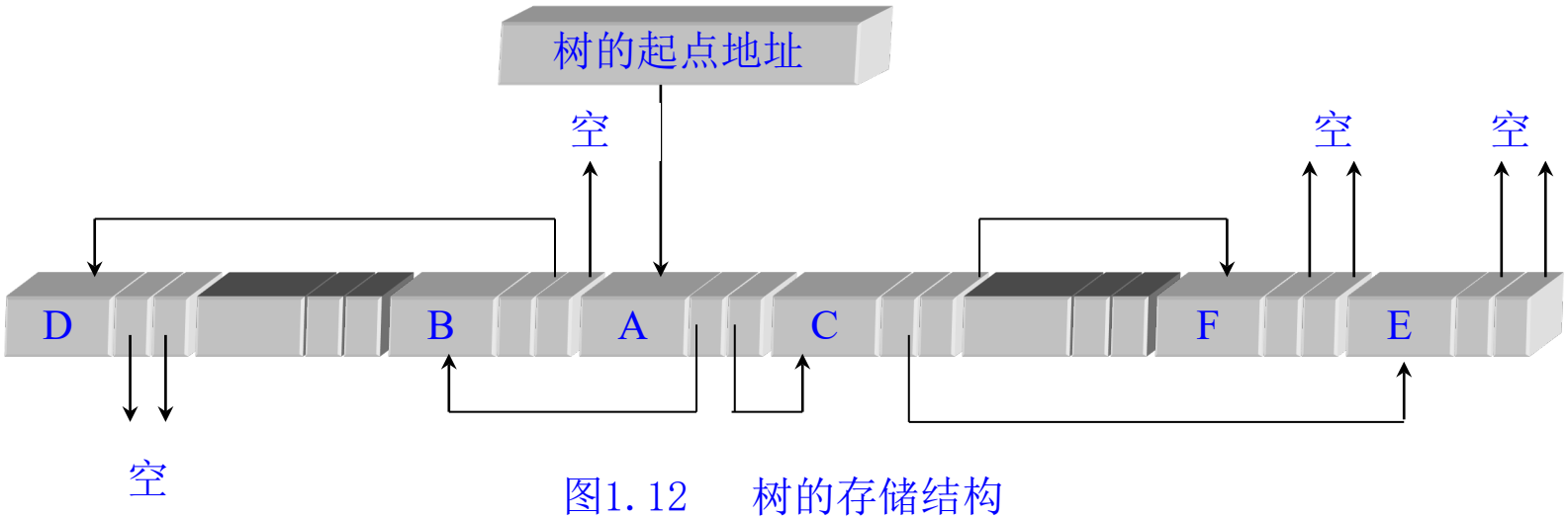


图1.12　树的存储结构

3. 数据域和链接域

1)数据域

数据域是物理存储空间中存储数据元素中数据值的空间。所占用的空间大小（字节数）依实际应用的数据元素中包含的信息量的大小而定。

如：成绩数据问题中，存储姓名和成绩两个数据项。

2)链接域

链接域又称**指针域**，是非顺序存储映象时表示数据元素之间关系的地址存储空间，是额外的空间付出。
所占用的空间大小（字节数）一般地与特定计算机的地址表示有关。

在顺序存储映象结构中，链接域是不存在的，数据元素之间关系是以物理存储的邻接方式隐含表示的。

1.2.4 静态存储空间分配和动态存储空间分配

存储管理是研究数据元素的空间分配和空间回收的方法和机制。

1. 静态存储空间分配和释放

静态存储空间分配：数据元素的分配（存储）过程是分两步完成：

第一步：为所有数据元素一次性获取连续的物理存储空间；

第二步：在获取的物理空间上对数据元素进行物理映象。

如：程序中需要使用数组

静态存储空间释放：所有空间使用完后才一次性释放，某个元素空间不使用时，并不释放。

2. 动态存储空间分配和释放

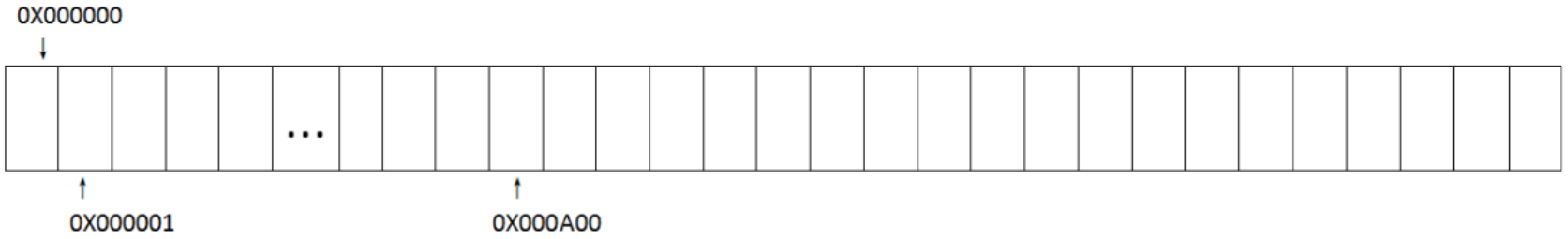
动态存储空间分配：数据元素的分配（存储）过程是一步完成：即，每次为一个数据元素获取单个物理存储空间，同时完成数据元素在物理空间上的映象。

如：在C++语言中利用new申请变量空间（动态空间申请），并将数据元素的值存储到该空间中。

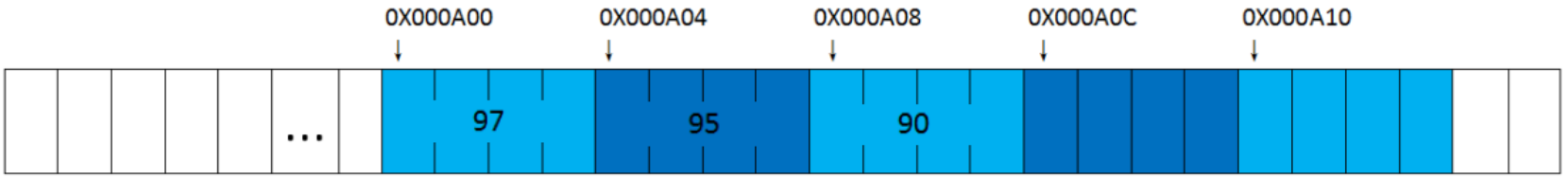
动态存储空间释放：某个数据元素空间不使用时，立即释放。

如：在C++语言中利用delete释放变量空间（动态空间释放，C++系统不会主动释放空间）。

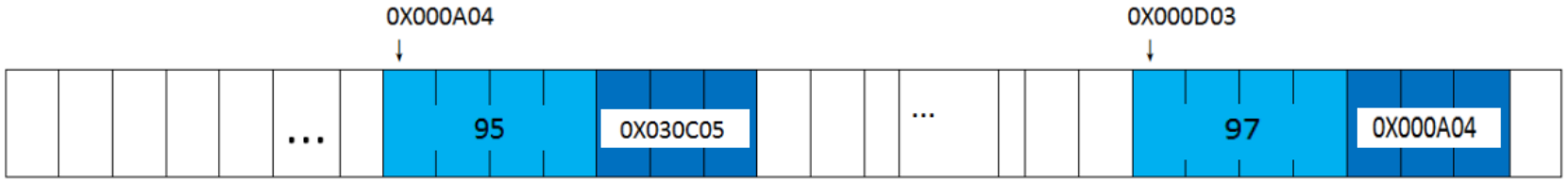
物理空间（线性特征）示意图（每个空间为 1 个字节）



物理空间:顺序存放结构示意图（数据元素值是 int）



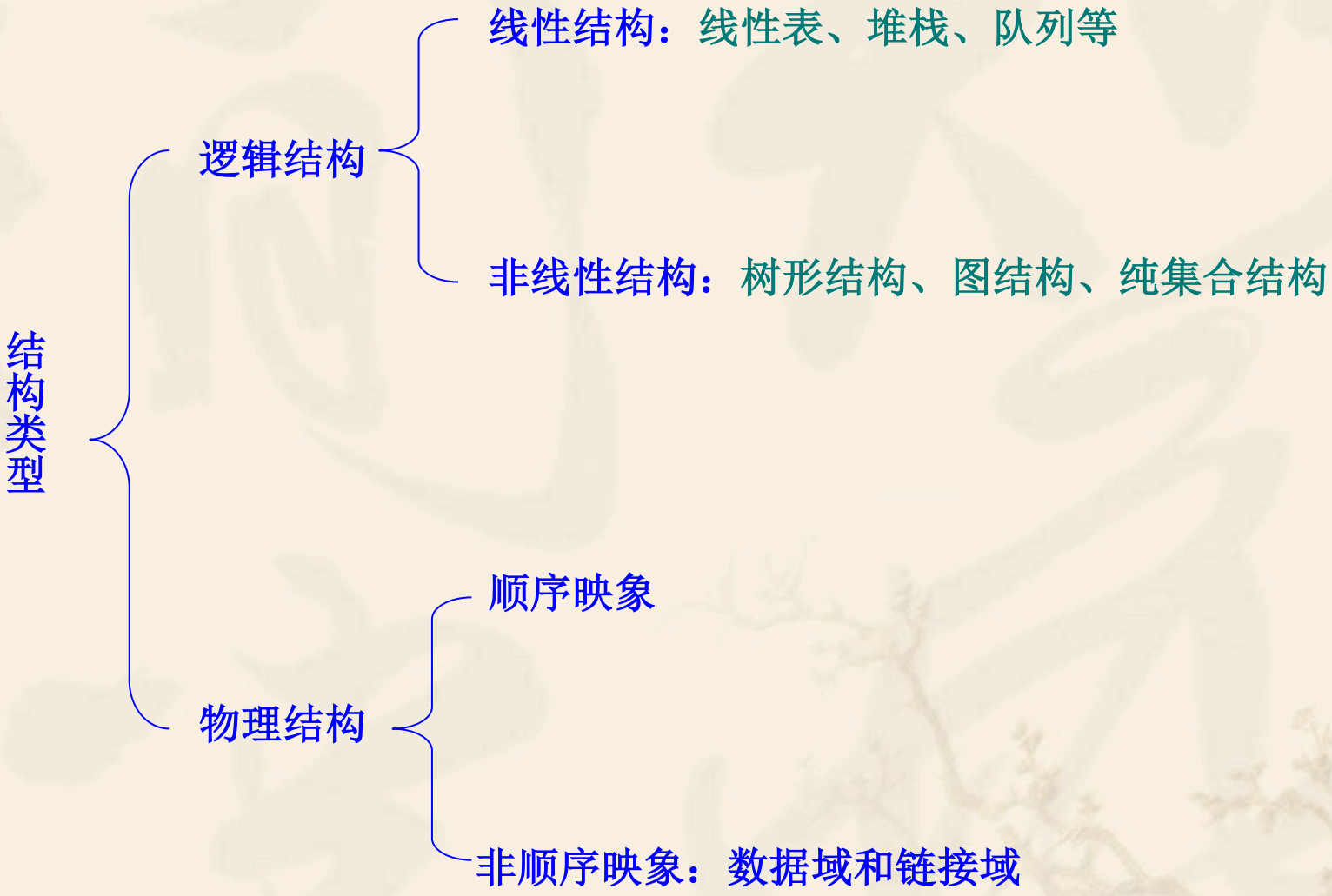
物理空间:非顺序存放结构示意图（数据元素值是 int）



在计算机系统中，操作系统、数据结构、高级语言等都涉及到物理存储空间的分配和回收，统称为存储管理。存储管理分为三个不同的层次:

- 进程所需要的存储空间由操作系统分配，一旦进程结束，操作系统就回收进程所使用的存储空间。
- 数据元素存储空间由某个进程进行分配或回收。例如编译进程为变量、数组等进行存储空间的分配和回收。
- 数据元素存储空间由数据结构管理系统进行分配或回收。这类存储空间的管理问题是数据结构研究的范畴。

1.2 数据结构的相关概念



1.3 数据类型、抽象数据类型和数据结构

1.数据类型

数据类型具体含义是，它描述了一组数据和在这组数据上的操作或运算及其操作或运算的接口。

2.抽象数据类型

抽象数据类型是指不涉及数据值的具体表示，只涉及数据值的值域，操作或运算与具体实现无关，只描述操作或运算所满足的抽象性质的数据类型和接口。

ADT 线性表的抽象数据类型

ADT LinearList
{

Data：有限元素（$e_0, e_1, \ldots, e_i, \ldots, e_{n-1}$）的有序序列的集合。

Relation：e_{i-1}是 e_i的直接前驱元素，e_i一定在元素e_{i+1}之前，e_{i+1}是e_i的直接后继元素，e_{i+1}一定在元素e_i之后。而且，每个元素只有一个直接前驱元素，也仅有一个直接后继元素。

Operation:

Creat(MaxListSize)	构造空线性表,其中有MaxListSize个空间
Destroy()	删除线性表
GetElement(k, result)	在线性表中，取第k个元素，存入result中
Search(searchkey)	在线性表L中，查找元素或元素关键字为searchkey的元素
Insert(k, new)	在线性表L中，第k个数据元素之后中插入数据元素new
Delete(k)	在线性表L中，删除第k个数据元素
Length()	求线性表长度
GetElementAddress()	获取线性表首地址
IsEmpty()	判断线性表L中有无元素

}

1.4 算法及算法分析、算法描述

1.4.1 算法和程序

1. 算法

算法：是指解决问题的一种方法或一个过程。算法可以理解为函数的另一种表述，所以，一个给定的算法解决一个特定的问题。

算法也可以描述为：算法是非空的、有限的指令序列，遵循它就可以完成某一确定的任务。

它有五大特征：

- 有穷性：一个算法在执行有限步骤之后必须终止。
  如果步骤是无限地，并无法全部写出不是算法。
- 确定性：一个算法所给出的每一个计算步骤（精确定义的，无二义性。
  确定性说明的是，算法中所执行下一步是确定的。
- 可行性：算法中要执行的每一个步骤都可以在有限时间内完成。
- 输入：算法有一组作为加工对象的量值，就是一组输入变量。
- 输出：算法有一个或多个信息输出，它是算法对输入量的执行结果。

2.算法与程序差异

程序：是使用某种程序设计语言对一个算法或多个算法的具体实现。

- 算法只是程序在的一部分，算法是程序解决问题的核心。
- 同一个算法，由于使用不同的设计语言实现，所以，可能有许多程序对应于同一算法。
- 算法必须提供足够的细节才能转化为程序。
- 算法是可终止的，但程序不一定，程序可在无外来干涉情况下一直执行下去；
- 程序可以既无输入数据，又无输出信息。

```
#include <iostream.h>

int factorial (int n); //这是向前引用的函数说明
int main()
{//这是主程序
    int             n=6;
    int             result;
    result= factorial (n);
    cout<<" result="<<result<<endl;
    return 0;
}

int factorial (int n)
{//这是算法（函数），是程序的一部分
        int y=1;
        for (int i = 1; i < n; i++)
    {
        y=y*i;
    }
    return y;
}
```

1.4.2 程序性能和算法效率

程序性能：是指运行一个程序所需在的时间多少和空间大小。

程序是由算法组成的，程序的好坏主要是由算法的优劣所决定的。

考虑以下**三个方面**：

- 依据算法所编制的程序在计算机中运行时占有内存空间的大小（也要考虑占辅存空间的多少），即**空间特性**。
- 依据算法所编制的程序在计算机中运算时所消耗的时间，即**时间特性**。
- 算法是否易理解，是否易于转换成任何其它可运行的语言程序以及是否易于调试。

1.空间复杂性

1）指令空间：

指令空间是指用来存储经过编译之后的程序指令所需的空间。指令空间一般不是数据结构所讨论的问题。

2）数据空间：

存储数据元素的空间及程序中工作变量，包括：

存储常量和简单变量所需要的空间

存储复合变量所需要的空间

数据元素值占用的空间（重要）

3）环境栈空间：

环境栈用来保存函数调用和返回时需要的信息。包括返回地址、局部变量的值、参数的值。调用或递归的层次越深，所需有环境栈空间就越大，这一部分的空间是可变部分。

（重要）

递归层次保留现场信息说明

Factorial(n) 保留现场信息n

Factorial(n-1) 保留现场信息n-1

Factorial(n-2) 保留现场信息n-2

.

.

Factorial(1) 保留现场信息1

Factorial(0)

2.时间复杂性

一个程序在计算机上运算所消耗的时间主要取决下述因素：

① 程序运行时所需要输入的数据总量消耗的时间。

② 对源程序进行编译所需要的时间。

③ 计算机执行每条机器指令所需要的时间。

④ 程序中关键指令重复执行的次数。

第四因素主要可从两个方面估算：

- 找出一个或多个关键操作
- 确定这些关键操作所需要的执行时间

关键操作：程序执行过程中，数据的**比较**、变量的赋值（**数据移动**）、过程或函数的**调用**都是程序的关键性步骤

执行时间：执行时间主要是由执行次数决定。确定程序执行步骤的次数，由其是执行**关键操作的次数**。

程序中影响次数的程序结构主要是**循环结构**。

1.4.3 算法分析

算法分析时只能分析算法中关键的部分，而忽略相对次要的算法操作步骤。

不同的“操作”的**难易程度**是不同的，如z1相对z2计算量较简单：

```
z1=y+10
z2=y*y+100
```

在算法分析时，每个都认为是“一个操作步骤”

语句频度即为“操作步骤”或“语句”重复执行的次数。

为描述算法的时间复杂性，将时间复杂性用函数T(n)表示，n表示算法中处理数据对象的数量或问题的规模的度量。

如：$T(n)= 2n^3+3n^2+2n+1$

当n足够大时，有$T(n)\approx 2n^3$，或者$g(n)=2n^3$

引入渐进符号“O”，用大写O字母表示函数T(n)的上限函数，即当n足够大时的函数，记为：

$O(g(n))= O(n^3)$，即 $O(n^3)$数据级

典型的复杂性函数的表示（a,b,c为已知数）：

- 常数函数：$O(g(n))= O(9+12)= O(1)$
- 线性函数：$O(g(n))= O(a*n+b) =O(n)$
- 对数函数：$O(g(n))= O((a*n*\log_2 n +b*n)= O(n*\log_2 n)$
- 平方函数：$O(g(n))= O(a*n^2+b*n)= O(n^2)$
- 指数函数：$O(g(n))= O(a^n + b*n^2+c*n)=O(a^n)$

常数函数是指算法的复杂性与算法中处理数据对象的数量（规模）无关，比如：

$T(n)= 15$

$T(n)= 20056$

不同级别的复杂性函数存在着以下关系：

$O(1)< O(\log_2 n)<O(n)< O(n*\log_2 n)< O(n^2)< O(a^n)$

例如，两个n×n的矩阵相乘

```
void matrix-product（int a[][]，int b[][]，int c[][],int n）；
 {
    for ( i=1;i<=n;i++ )                          //重复次数:n+1
      for ( j=1;j<=n;j++ )                        //重复次数:n(n+1)
        {
          c[i][j] =0;                             //重复次数:n²
          for ( k=1;k<=n;k++ )                    //重复次数:n²(n+1)
            c[i][j] =c[i][j]+a[i][k]*b[k][j];     //重复次数:n³
        }
 }
```

算法执行时间的度量，记作：

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

三个简单的程序段:

1. x=x+1; //语句频度为 1

2. for (i=1;i<=n;i++)
   { x=x+1;} //语句频度为n

3. for (i=1;i<=n;i++)
   for (j=1;j<=n;j++)
   { x=x+1;} //语句频度为n*n

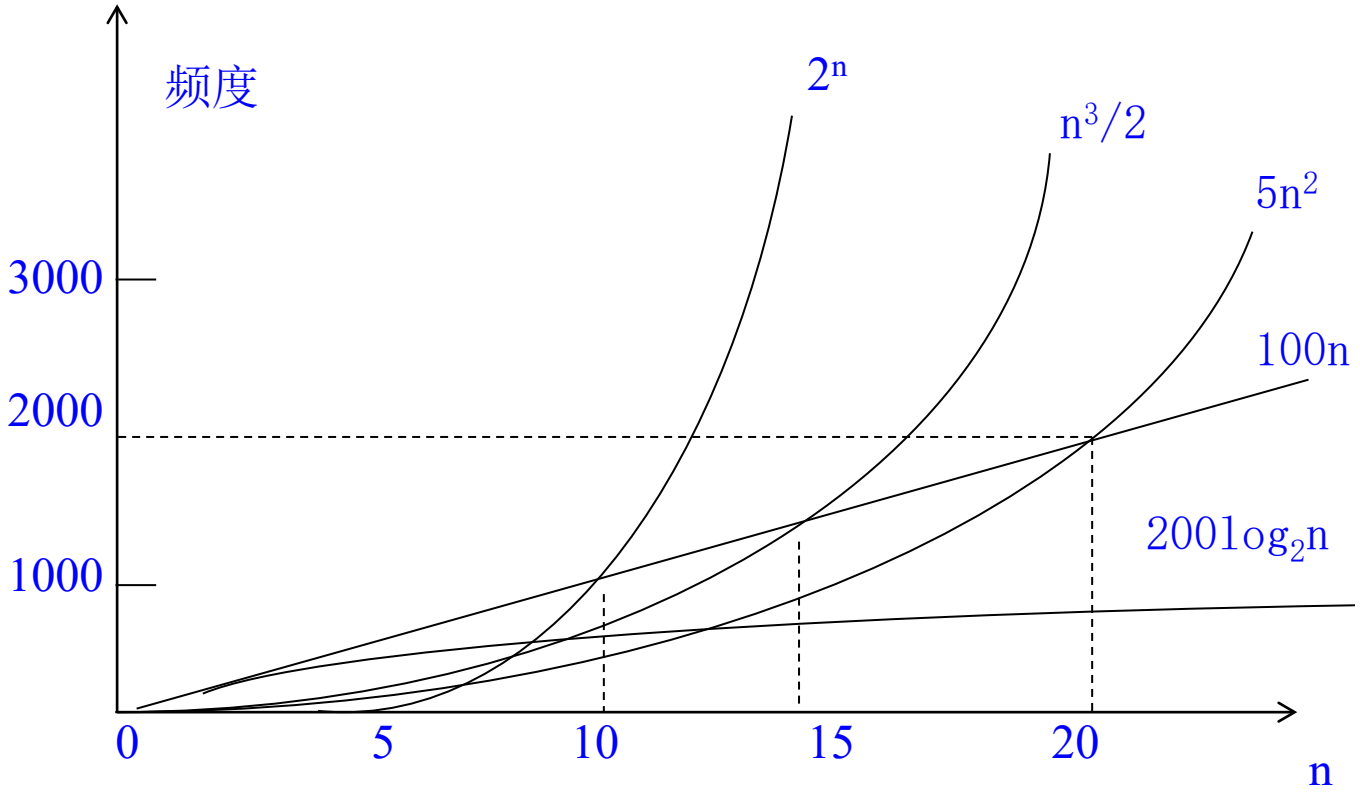


图1.4.1　各种数量级的T(n)

表1.4.1　在某一时间内不同算法所能处理的输入量上限

算法	时间复杂性	1秒钟能处理 的最大输入量	1分钟能处理 的最大输入量	1小时能处理 的最大输入量
A1	n	1000	6×10^4	3.6×10^6
A2	$n\log_2 n$	140	4893	2.0×10^5
A3	n^2	31	244	1897
A4	n^3	10	39	153
A5	2^n	9	15	21

表1.4.2　计算机速度提高10倍和1万倍的效果

算法	时间复杂性	提高前单位时间内能处理的最大输入量	提高10倍后单位时间内能处理的最大输入量	速度提高1万倍后单位时间内能处理的最大输入量
A1	n	S1	10 S1	10000 S1
A2	$n\log_2 n$	S2	对大的S2接近于S2	当$10\log_2 S2 > \log_2 9000$时 超过9000S2
A3	n^2	S3	3.16 S3	100 S3
A4	n^3	S4	2.15 S4	21.54 S4
A5	2^n	S5	S5+3.32	S5+13.32

1.4.4 算法描述

描述算法通常采用一种高级语言来描述，本书中将用类C++语言（C++的一个子集）来描述。规定如下：

1.数据元素结构的表示用类型定义

方式一：使用C++结构类型定义数据元素描述如下：

```
struct ElementType
{
    Item1Type       item1;
    Item2Type       item2;
        …
    Item3Type       item3;
};
```

C++结构类型定义时，结构成员默认是全局变量（public）

方式二：

使用C++类定义数据元素描述如下：

```
class ElementType
{
    Item1Type          item1;
    Item2Type          item2;
        …
    Item3Type          item3;
};
```

C++类定义时，结构成员默认是私有变量（private）

2.算法函数的描述

1）输入语句

cin>>输入变量;

如输入一数据到searchkey中：

```
cin>>searchkey;
```

2）输出语句

cout<<输出变量;

如输出searchkey的值:

```
cout<<searchkey<<endl;
```

3）赋值语句

变量=表达式;

4) 条件语句

```
if (条件表达式)
 {
     <语句序列>
}
```

或者

```
if (条件表达式)
 {
     <语句序列1>
}
else
 {
     <语句序列2>
}
```

5）循环语句

```
while  (条件表达式)
 {
     <语句序列>
}
或
do
 {
     <语句序列>
} while  (条件表达式)
或
for(循环变量=初值; 条件表达式;步长表达式)
 {
     <语句序列>
}
```

6）返回语句

return （返回表达式）;

7）定义函数

（1）一般C++函数定义：

```
<函数类型> <函数名> (函数形式参数表)
{ //算法说明
    <语句序列>
}
```

（2）C++模板函数定义：

```
template<class T>
<函数类型> <函数名> (函数形式参数表)
{ //算法说明
    <语句序列>
}
```

（3）C++模板类函数定义：

```
template<T>
<函数类型> <类名><T>:: <函数名> (函数形式参数表)
{  //算法说明
     <语句序列>
}
```

```
template<class  T>
bool LinearList< T>::InsertElementLinearList( int k,  T &newvalue )
{
    if (k < 0 || k > length )              return false;
    if (length == MaxSpaceSize)    return false;
    for (int i = length-1; i >= k; i--)
       element[i+1] = element[i];
    element[k] = newvalue;
    length ++;
    return true;
}
```

3. 内存空间的动态分配和释放

C++风格的分配空间和释放空间描述如下：

1）分配空间：指针变量=new 数据类型;

如要申请一个int类型的空间并由p指针，则申请语句：

```
p=new int;
```

2）释放空间：delete 指针变量;

如要释放一个p所指的空间，则释放语句：

```
delete p;
```

课程难点

- 递归思维与递归程序设计
- 指针、地址概念与使用
- 一些基本概念：
  - 组织结构、逻辑结构、物理结构
  - 操作、计算、元组、……

尽快进入角色

- 尽快学会一种可以工作的语言以及软件开发工具
- 会图像界面程序设计
- 软件应用开发平台和工具都有哪些 (baidu.com)
- 开发平台_百度百科 (baidu.com)
- 软件开发平台有哪些 (sg-info.cn)
- 集成开发环境_百度百科 (baidu.com)

软件开发

- 桌面软件开发
- 服务器软件开发
- 网页开发
- 科学计算

谢　谢！