侯风巍 编著
杨永田 主审

数据结构要点精析

——C语言版（第2版）

北京航空航天大学出版社

数据结构要点精析

——C 语言版(第 2 版)

侯风巍　编著
杨永田　主审

北京航空航天大学出版社

内 容 简 介

本书介绍数据结构线性表、栈和队列、串、数组和广义表、树和二叉树、图、查找、内排序等的基本概念、基本知识点、相关结论和各种数据类型的不同存储结构以及主要操作的实现算法；系统而全面地对读者在学习过程中可能遇到的问题，在相应的知识点处提出并加以解决；精选各大知名院校和研究所的硕士研究生入学试题及国内外教材中有代表性的习题，结合各相关知识点进行深入细致的分析、完整的解答和点评扩展。

本书可作为计算机专业本、专科学生的教学参考书，也可作为报考计算机专业硕士研究生的学习参考书，还适于计算机等级考试者及广大工程技术人员和自学者参考。

图书在版编目(CIP)数据

数据结构要点精析：C语言版/侯风巍编著. —2版.
北京：北京航空航天大学出版社，2009.3
ISBN 978-7-81124-426-7

Ⅰ.数… Ⅱ.侯… Ⅲ.①数据结构—高等学校—教学参考资料②C语言—程序设计—高等学校—教学参考资料
Ⅳ.TP311.12 TP312

中国版本图书馆CIP数据核字(2008)第096722号

数据结构要点精析——C语言版(第2版)

侯风巍 编著
杨永田 主审
责任编辑 宋淑娟

*

北京航空航天大学出版社出版发行
北京市海淀区学院路37号(100191) 发行部电话：010-82317024 传真：010-82328026
http://www.buaapress.com.cn E-mail:bhpress@263.net
北京时代华都印刷有限公司印装 各地书店经销

*

开本：787×1 092 1/16 印张：23.25 字数：595千字
2009年3月第1版 2009年3月第1次印刷 印数：4 000册
ISBN 978-7-81124-426-7 定价：35.00元

第2版前言

自从本书于2007年出版发行以来，受到了广大读者的关心和推崇。在本书的伴随下，很多读者也已磨刀霍霍向牛羊，真正体验到了学习数据结构的从容与快乐。这的确是一件值得弹冠相庆的事情，这也正是编写本书的初衷之所在，故而编者感到莫大的光荣和欣慰。

事情总是在发展的，本书也不能停留在同一个位置上。随着学科发展及应用范围的进一步拓宽和加深，对数据结构知识点的应用和考查也有所变化。这就要求本书的内容必须适应和反映其中的变化，并与时代的发展保持同步；同时本书力求做到保持“解牛之道”不变，以不变应万变。

本着对数据结构之道不懈追求的态度，编者在第1版的基础上对本书内容进行了修订、调整和扩充。

一方面，在保持第1版整体结构不变的情况下，对各章节内容进行了全面扩充和修正，增加了各大高校和科研院所近几年硕士研究生入学考试中的经典题目，并进行了详细而全面的解析。大家都知道，剖析、理解经典题目是掌握相应知识点的捷径，这也正是本书一直坚持使用考研真题作为解析知识点的原因所在。同时增加了链表、栈、树、图、排序中的一些必要知识点，并以联想网络的方式与原有知识网有机结合起来。无论是对题目的解析还是对知识点的诠释，本版都试图做到尽可能细致而全面。天下难事必做于易，天下大事必做于细。如果对每个知识点、每道题目都能钻研到细致入微处，那么掌握数据结构这门学科并游刃有余地应用数据结构知识解决实际问题，自然也就变得容易实现多了。

另一方面，在改正第1版中发现的错误的基础上竭尽全力避免在新增加的内容中再引入新的错误。然而，由于编者水平有限，本版中某些欠缺和不妥之处仍会在所难免，敬请广大读者继续提出宝贵意见和建议。

本书不是读完一遍就可以束之高阁的快餐读物，也不是能够立刻解决任何问题的万能题库，而是需要各位读者反复阅读体会，把“解牛之道”极力融入自己思想之中，这样才能真正达到“恢恢乎其于游刃必有余地”的境界。希望这本书能够帮助各位读者跨越数据结构的重重障碍，在高处领略“提刀而立，为之四顾，为之踌躇满志”的壮美。

侯风巍

2008年6月22日

前　言

《庄子·养生主》曰："彼节者有间，而刀刃者无厚。以无厚入有间，恢恢乎其于游刃必有余地矣。"

数据结构在大多数人眼中可能是一头比较难解的"牛"，然而这头全牛也必节之有间。编者愚钝窃以为略微品得个中滋味，且从中获益，料想一定也有很多人能够从中获益，于是不敢独享，遂编著此书以与各位读者共享。编者利用一把无形之刀刃入数据结构基本知识点、基本原理之"间"，淋漓尽致地展现其动人面貌；同时又能恰当地从整体的角度发散、拓展开来，将各相关知识连接成网，为读者建立起一张秩序井然的知识结构网络。读过本书后，读者或许会踌躇满志，大发感慨：目无全牛矣。

通过阅读本书，读者能够潜移默化地学到应对具体问题的分析方法和解决方法的原则以及在实战中锻炼处理问题的能力，能够举一反三，尽量游刃于各"节"之"间"。在讲述知识的同时适当地提出问题，使读者不是被动地看书，而是积极地参与进来，有机会牛刀小试，增强了互动性。

精心选择的例题构成了本书特色之一，一道好的题目能够从各个方面反映出尽可能多的知识点。各大知名高校的研究生入学考试题（在所选例题处注明）不可谓不精到，越是做这些经典题目，越能让我感觉到数据结构及其"解牛刀"的奥妙之所在。

算法程序即彼节者，本亦有间，编者适度的注释更增其"间"数，使其架构清晰明朗；朗朗上口的谐音、口诀，独特的技巧、方法更使本书生趣盎然；二叉排序树的平衡处理、广义表的头尾分解图……这些独到的方法以条例步骤的方式将其中之"道"诠释得更是淋漓尽致，使读者轻松自如地游刃其中。

在细微处下工夫，能够恰到好处地对容易误解或混淆的知识点、概念、结论、疑点等加以剖析，并给出正确的结论和判断，由小见大；在讲述算法过程中，并不仅仅着眼于介绍某种算法，而是强调如何传递一种分析问题和解决问题的方法，用来解眼前的"牛"，从而引出某种算法，并对算法的优缺点加以分析，以及总结适用于何种场合等，还从时间复杂度和空间复杂度等角度对算法加以分析，提出修改的建议和方案。本书力求做到不仅知其然，还知其所以然，更要做到恢恢乎游刃有余。

图释与言解的有机结合是本书的另一特色，其中利用大量美观的连续图表来诠释基本思想，将平衡树的平衡处理用生动的组图表现得活灵活现；把最小生成树的生长过程像播放动画片一样展现出来，呼之欲出；使堆栈、队列的变幻莫测跃然纸上，引人入胜。

本书所有程序均采用C语言编写。书中所有符号均使用正体，以达到与算法对应及全书体例上的统一。另外，因书中荟萃了大量不同学校的考试题目，为了尽量与原题保持一致，所以对可能存在的不一致的表达形式未做改动。

本书凝聚了许多数据结构方面文献的精华和老师曾对我传授的“道”，以及编者多年来对数据结构科学探索研究所取得的经验与心得。书中的题目来源于各大知名院校研究生入学考试试题和一些教材、参考书，在此一并表示感谢。

本书由哈尔滨工程大学计算机科学与技术学院的博士生导师杨永田教授审定，他花费了大量时间仔细审阅了全部内容，无论从宏观把握还是在微观考究上都提出了许多宝贵意见和建议，他广博的学识和严谨的治学态度及其重要的意见和建议对本书的质量起到十分重要的保障作用，在此表示衷心感谢。

焦金良、李东勤、李长城为本书提供了许多宝贵资料和建设性意见；吕金锁、张斌等为本书提供了许多建设性意见，在此对他们的热心帮助表示感谢。

在此尤其要感谢我的父母对我的默默奉献与支持。

由于编者水平有限，编写时间仓促，其中定有疏忽或错误之处，敬请广大读者批评指正，提出宝贵意见和建议。联系邮箱：hfw_book@yahoo.com.cn。

侯风巍

2006年8月

目　录

第1章 绪 论

【学习要点】

1. 理解数据、数据对象、数据元素和数据结构等基本概念，尤其是数据的逻辑结构与物理(存储)结构间的关系以及在这种结构上所定义的操作。

2. 掌握算法的定义和特性、算法的时间复杂度和空间复杂度。

3. 掌握计算语句频度和估算算法的时间复杂度和空间复杂度的方法。

【要点精讲】

本章主要讨论数据结构学科的基本概念及其所研究的主要内容，包括算法的概念、特点、要求及其评价方法。

要使用计算机解决现实世界中的问题，就需要利用一些数据结构来表达现实生活中的各种事物，进而对实际问题进行建模，并加以解决。大体上数据结构可分为逻辑结构和物理结构，而逻辑结构又可分为线性结构和非线性结构。算法和程序是不同的，程序是用某种计算机语言实现了的算法，而算法是更高层次上的抽象。

在各种类型的考试中，比较侧重于对数据结构、数据类型、ADT 和算法等重要基本概念的考察，对算法的描述方法以及评价标准与方法的考察，也请读者特别注意。

1.1 基本概念

1. 数据(data)

数据是信息的载体，是对客观事物的符号表示，是所有能输入到计算机并被计算机程序处理的符号总称。

2. 数据元素(data element)

数据元素是数据的基本单位。

3. 数据项(data item)

数据项是数据不可再分割的最小单位。

注意：数据元素和数据项的区别

数据元素一般在计算机程序里被看做一个整体来考虑和处理。一个数据元素可以是不可分割的原子，也可以由若干个数据项组成。数据项强调不可再分性。

4. 数据对象

数据对象是性质相同的数据元素的集合。

5. 数据结构

数据结构是具有特定关系的数据元素的集合。主要研究数据的逻辑结构(数据元素之间的逻辑关系,它与所使用的计算机无关)和物理结构(数据结构在计算机中的存储表示,既包括数据元素在计算机中的存储方式,也包括数据元素之间的逻辑关系在计算机中的表示,因此它依赖于计算机)以及施加于该数据结构上的操作及其相应算法并评价算法的优劣。

1.1.1 数据的逻辑结构

数据的逻辑结构指各数据元素之间的逻辑关系,是用户按使用需要建立的,与数据元素本身的内容和形式无关,与所使用的计算机无关。根据数据元素之间的不同关系分为:集合、线性结构、树形结构、图状结构或网状结构等 4 种。其中后两种又被称为非线性结构。因此可以说,数据的逻辑结构分为线性结构和非线性结构两大类。

线性结构的特点是:所有数据元素都处于一个序列中,有且只有一个开始元素和一个终端元素,所有元素最多有一个直接前驱和一个直接后继。

非线性结构的特点是:一个数据元素可能有 0 个、1 个或多个直接前驱和直接后继。

数据的逻辑结构在形式上用一个二元组表示,即

$$B=(D,S)$$

其中,D 是数据元素的有限集,S 是 D 上的关系的有限集,而每个关系都是从 D 到 D 的关系。设 s 是一个从 D 到 D 的关系,$s\in S$,若 $d,d'\in D$,且$\langle d,d'\rangle\in s$,则称 d'为 d 的后继,d 是 d'的前驱,这时 d 和 d'是相邻(相对于 s 而言)的结点;如果不存在一个 d'使$\langle d,d'\rangle\in s$,则称 d 为 s 的终端结点;如果不存在一个 d'使$\langle d',d\rangle\in s$,则称 d 为 s 的开始结点;如果 d 既不是终端结点也不是开始结点,则称其为内部结点。

Example 1-1

下列是几种用二元组表示的数据结构,画出它们分别对应的逻辑图形,并指出分别属于何种结构。(这里的圆括号对表示两个结点是双向的)

① M=(D,S),其中

$$D=\{a,b,c,d,e,f,g,h\}$$

$$S=\{\langle d,b\rangle,\langle d,g\rangle,\langle d,a\rangle,\langle b,c\rangle,\langle g,e\rangle,\langle g,h\rangle,\langle e,f\rangle\}$$

② N=(D,S),其中

$$D=\{1,2,3,4,5,6\}$$

$$S=\{(1,2),(2,3),(2,4),(3,4),(3,5),(3,6),(4,5),(4,6)\}$$

【解】

这道题重在考察对基本概念的理解以及对数据的逻辑结构二元组表示方法的把握和理解。

① M 对应的逻辑图形如图 1-1 所示,它是一种非线性结构中的树形结构。

由图 1-1 可以看出,d 为开始结点(无前驱的结点),a,c,f,h 为终端结点(无后继的结点)。

② N 对应的逻辑图形如图 1-2 所示,它是一种非线性结构中的图形结构。

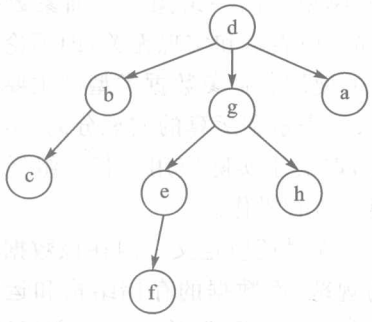

图 1-1 M的逻辑图

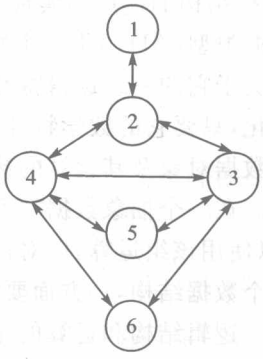

图 1-2 N的逻辑图

1.1.2 数据的存储结构

数据结构、数据元素、数据项在计算机中的表示分别称为存储结构、结点和数据域。

数据结构有两种不同的存储结构：顺序存储结构和链式存储结构。顺序存储结构是借助数据元素在存储时的相对位置来表示数据元素间的关系，而链式存储方式则是用指示元素存储地址的指针来表示元素间的逻辑关系。

1.1.3 数据的逻辑结构与存储结构的关系

数据的同一逻辑结构可以对应不同的物理结构。算法的设计依赖于数据的逻辑结构，而算法的实现则取决于特定的物理结构。

研究数据结构是为了解决应用问题，所以讨论数据结构必须同时讨论在数据结构上执行的相关运算及其算法才有意义。通过对运算及其算法的性能分析和讨论，使在求解应用问题时，能选择和设计适当的数据结构，编写出高效的程序。

1.2 抽象数据类型

数据类型指一个类型及定义在这个类型上操作的集合。

通常将数据和操纵数据的运算组成一个独立的子函数，每个子函数都有一个明确定义的接口，模块内部信息只能经过这一接口被外部访问。以后在使用此函数时，只须了解其所能完成的功能即可，而无须具体了解其内部实现机制、实现过程及内部所使用的数据结构的具体实现方式。这就实现了信息隐藏，具体有以下几个优点：

① 开始时完成此项功能使用了一种方式，获得一些经验后，可能发现另一种方式会更好，这时可以对子函数体进行优化修改，但不改变接口定义。如果每次都将实现这种功能的函数操作写出来，则需要在每一次出现这些指令的地方都做修改。如果采用调用独立子函数操作的方法，则仅须修改函数的声明即可。

② 在阅读程序时，若出现子函数名称，就能立即明白程序所要进行的具体操作，虽然可能暂时还不清楚具体指令。

③ 将数据结构的使用与实现分离,有助于自顶向下地设计数据结构和程序。

抽象数据类型(ADT)指一个数学模型以及定义在该模型上的一组操作。抽象数据类型的定义仅取决于它的一组逻辑特性,而与其在计算机内部如何表示和实现无关,即不论其内部结构如何变化,只要它的数学特性不变,都不影响其外部的使用。抽象数据类型的主要特征包括该类型的数据对象及其运算的接口规范,它与数据对象的表示和运算的实现分离,实行封装和信息隐蔽。在一个抽象数据类型上应当定义哪些运算,取决于实际应用。若一组运算是完备的,就可以使用该组运算,以对该数据结构实现所希望的所有操作。

对于一个数据结构,一方面要说明其数据的逻辑结构,同时还应定义一组在该数据结构上执行的运算。逻辑结构和运算的定义组成了数据结构的规范,而数据的存储结构和运算算法的描述构成了数据结构的实现。规范是实现的准则和依据,它指明“做什么”,而实现解决“怎样做”。从规范和实现两方面来讨论数据结构的方式是抽象数据类型的观点。在本书中,一种数据结构被作为一个抽象数据类型来加以讨论。

注意:

数据结构定义了一组按某些关系结合在一起的数据元素;数据类型不仅定义了一组带结构的数据元素,而且还在数据元素上定义了一组操作。

1.2.1 算 法

算法是对特定问题求解步骤的一种描述,是指令的有限序列。它有 5 大特性:有穷性、确定性、可行性、有输入、有输出。

① 有穷性:算法总能在执行有限步之后终止。

② 确定性:算法的每一条指令都有确切的含义,没有二义性。

③ 可行性:算法的每一条指令都可以通过执行有限次已经实现的基本算法来实现。

④ 有输入:算法有零个或多个输入。

⑤ 有输出:算法至少产生一个输出。

记忆方法 “出”“入”“可”“确”“穷”,谐音为“出入可去穷”。

评价一个算法一般从以下几个方面进行:正确性、可读性、健壮性、效率(时间复杂度)与低存储量需求(空间复杂度)。

① 正确性:算法的执行结果应当满足预先规定的功能和性能要求。

② 可读性:一个算法应当思路清晰、层次分明、简单明了、易读易懂。

③ 健壮性:当输入不合法数据时,算法应能做出适当的处理,而不至于引起严重的或莫名其妙的后果。

④ 效率与低存储量:尽可能地降低时间消耗,减少辅助空间量。

算法的正确性指在合法的输入下,算法能实现预先规定的功能和计算精度要求。而算法的健壮性则要求程序万一遇到意外时,能按某种预定的方式做出适当的处理。正确性和健壮性是相互补充的。正确的算法并不一定是健壮的,而健壮的算法又不一定是绝对正确的。

辨析 **算法就是程序?**

算法是程序的说法是不正确的。

第一,算法有 5 大特性:有输入、有输出、确定性、有穷性、可行性。程序可以不满足有穷

性，例如：一个操作系统，在用户未使用前一直处于“等待”的循环中，直到出现新的用户事件为止。这样的系统可以无休止地运行，直到系统停工。

第二，算法是面向功能的，可以用面向过程的方式描述；而程序可以用面向对象的方式描述。

第三，算法重在描述功能，传递的是一种思想，而不拘泥于使用哪种语言，甚至可以使用自然语言；而程序则是算法的具体实现，是用某种特定语言实现的，这种语言是机器可以识别的语言。

1.2.2 算法的分析

算法分析的目的是分析算法的效率和空间占用量，以求改进。估算算法运行时间的基本考虑是：撇开所有与计算机硬件和软件相关的因素，只确定依赖于问题的规模（待处理数据的数量 n）和确定算法执行的基本操作（某个数据类型上的标准操作）的次数。一个算法由顺序、分支和循环 3 种控制结构和原操作（固有数据类型的基本操作）构成，则算法取决于两者的综合效果。算法分析的两个主要方面是时间复杂度和空间复杂度。

1. 时间复杂度

由于估算算法的时间复杂度所关心的只是算法执行时间的增长率而非绝对时间，所以可以从算法中选取一种对于所研究的问题来说是“基本操作”的原操作，并以其重复执行的次数作为时间复杂度的依据。时间复杂度是一个数量级而不是一个绝对的时间常量，算法的时间复杂度取决于问题的规模 n 和待处理数据的初态。为了便于比较同一问题的不同算法，通常的做法是以上述依据作为算法运行时间的度量；同时，以最坏情况下的时间复杂度作为算法的时间复杂度。

以下为几种常用时间复杂度的比较：

$$O(1)<O(\text{lb } n)<O(n)<O(n\times \text{lb } n)<O(n^2)<O(n^3)<O(2^n)$$

语句的频度指某语句重复执行的次数。

Example 1-2

分别计算以下两个算法的时间复杂度：

①

```
I=1;
while(I<=n)
  I=I*2;
```

②

```
I=0;  s=0;
while(s<n)
  {I++; s=s+I;}
```

【解】

① 基本操作为乘法操作。只要确定该操作的重复执行次数，便可确定其数量级，从而该算法的时间复杂度也便得知。而重复执行次数取决于 I 值和 N 值的大小关系。

设重复执行的次数为 m，则 $I=2^m\leqslant n$，由此解出 $m=\text{lb } n$。因此，该算法的时间复杂度为 $O(\text{lb } n)$。

② 该算法的基本操作为 I++和 s=s+I。不难看出，s 为 I 在不同时刻所取值的累加和。设基本操作重复执行的次数为 m，则

$$s=1+2+3+\cdots+m=m\times(m+1)/2<n$$

得出 m=sqrt(2×n+1/4)-1/2[sqrt()为取平方根运算]。于是,本算法的时间复杂度为O[sqrt(n)]。

结语: 由此,读者不难体会到语句的执行频度和算法的时间复杂度之间的区别了。

2. 空间复杂度

算法所需要的存储空间分为三部分:输入数据所占用的空间、程序代码所占用的空间和辅助变量所占用的空间。一般,输入数据所占用的空间与算法无关,取决于问题本身;程序代码所占用的空间对不同算法不会有数量级的差别。所以,在估算算法的空间复杂度时,主要考虑算法执行过程中辅助变量所占用的空间。一般以最坏情况下的空间复杂度作为算法的空间复杂度。

Example 1-3

选择解决某种问题的最佳数据结构的标准是什么?

【解】

选择的标准是:

① 所需的存储空间量。

② 算法所需要的时间。而算法所需要的时间又包括以下几点:

- 程序运行时所需要的数据总量。
- 源程序进行编译所需要的时间。
- 计算机执行每一条指令所需要的时间。
- 程序中指令重复执行的次数,而本条正是讨论算法的重点内容。

Example 1-4(北京航空航天大学)

有实现同一功能的两个算法A1和A2,其中A1的时间复杂度为$T1=O(2^n)$,A2的时间复杂度为$T2=O(n^2)$,仅就时间复杂度而言,请具体分析这两个算法哪一个好。

【解析】

令$n^2=2^n$,可得到$n_0=4$。而当$n\geqslant n_0$时,$2^n\geqslant n^2$,于是可以认为最终A1的时间复杂度T1要比A2的时间复杂度T2大。仅就时间复杂度而言,算法A2更好一些。

注:

在设计算法时,一般情况下应尽可能避免使用时间复杂度是指数阶的,而应使用复杂度为低阶多项式的,以便能在有效的时间内解决问题。但是,虽然一般情况下时间复杂度为多项式的算法优于指数阶的算法,而时间复杂度为高次多项式的算法在问题规模n的很大范围内都不如某些时间复杂度为指数阶的算法,例如具有相同功能的两个算法其时间复杂度分别为$O(2^n)$和$O(n^{10})$,显然复杂度为指数$O(2^n)$的算法更优一些。

Example 1-5(华中理工大学)

调用下列C函数f(n)回答下列问题:

① 试指出f(n)值的大小,并写出f(n)值的推导过程;

② 假定n=5,试指出f(5)值的大小和执行f(5)时的输出结果。

C函数如下。

```
1. int f(int  n)
2. { int i,j,k,sum=0;
3.   for(i=1; i<n+1;i++)
4.   {for(j=n;j>i-1; j--)
5.     for(k=1;k<j+1;k++)
6.       sum++;
7.     printf("sum=%d\n",sum);
8.   }
9.   return(sum);
10. }
```

【解析】

① 该程序的主体由3层for循环构成。第5行for循环的功能是sum自增j−1+1=j;对于第4行的for循环而言,每进入一次循环体,sum就要自增j,而该循环要重复执行(n−i+1)次,分别使sum自增n,n−1,n−2,…,i(即j的各个取值),于是执行完第4行的for循环的结果为:使sum自增$i+(i+1)+\cdots+n=\frac{(i+n)(n-i+1)}{2}$;第3行的for循环使i从1增到n,每进入一次循环体就要使sum自增$\frac{(i+n)(n-i+1)}{2}$。于是,3层for循环可等价成下面语句:

```
for(i=1; i<n+1;i++)
  sum+=((i+n)(n-i+1))/2;
```

更为具体的,令n=5并调用函数f(n),在i从1增到5的过程中,sum自增的情况如表1-1所列。

表1-1 Example 1-5解表

i	1	2	3	4	5
sum自增量	1	2+1 + 2	3+2+1 + 3+2 + 3	4+3+2+1 + 4+3+2 + 4+3 + 4	5+4+3+2+1 + 5+4+3+2 + 5+4+3 + 5+4 + 5

不难得出:f(1)=1;f(2)=2×2+1=2^2+f(1)=5;f(3)=3×3+f(2)=14;f(4)=4×4+f(3)=30;f(5)=5×5+f(4)=55。

由此,可以推断$f(n)=n^2+f(n-1)$,f(1)=1,其中n≥1。

现在用归纳法证明该推断:

ⓐ 当n=1时,f(1)=1,显然结论成立;

ⓑ 假设n=k(k≥1)时结论成立,即$f(k)=k^2+f(k-1)$,其中sum的自增量为