21世纪高等职业教育主干课程辅导丛书

数据结构知识点与典型例题解析

陈亦望 主 审
杨 明 主 编
吉根林 严云洋 副主编

注重基础，答疑解惑
精选例题，分析透彻

清华大学出版社

21世纪高等职业教育主干课程辅导丛书

数据结构知识点与典型例题解析

陈亦望　主　审

杨　明　主　编

吉根林　严云洋　副主编

清华大学出版社

北　京

内 容 简 介

本书是结合数据结构主流教材，指导学生学习、练习及考试的辅导用书。本书共分9章，每章按其内容和教学进度分为若干节(大部分小节分两个板块进行讲解：基本知识点和典型题分析)。此外，本书后的附录中给出了期中、期末和专升本试题及其答案，以便读者在学习完本书后进行整体的测试。

本书可作为相关高校数据结构课程的参考书，也可作为相关考试(自学考试：专升本、程序员 、计算机等级考试(三级)等)的辅导用书。

图书在版编目(CIP)数据

数据结构知识点与典型例题解析/陈亦望主审；杨明主编；吉根林，严云洋副主编.
—北京：清华大学出版社，2005.9
(21世纪高等职业教育主干课程辅导丛书)
ISBN 7-302-08883-7
Ⅰ. 数… Ⅱ. ①陈… ②杨… ③吉… ④严… Ⅲ. 数据结构—高等学校：技术学校—教学参考资料
Ⅳ. TP311.12

中国版本图书馆CIP数据核字(2005)第084343号

出 版 者：清华大学出版社　　地　　址：北京清华大学学研大厦
　　http://www.tup.com.cn　　邮　　编：100084
　　社 总 机：010-62770175　　客户服务：010-62776969
组稿编辑：章忆文
文稿编辑：张　莉
封面设计：隋刘源
排版人员：王　婷
印 刷 者：北京市清华园胶印厂
装 订 者：三河市李旗庄少明装订厂
发 行 者：新华书店总店北京发行所
开　　本：185×260　印张：16.75　字数：399千字
版　　次：2005年9月第1版　2005年9月第1次印刷
书　　号：ISBN 7-302-08883-7/TP·6291
印　　数：1～4000
定　　价：25.00元

丛 书 序

随着高等教育大众化的推进，高等职业教育在我国的高等教育领域内已占有半壁江山。与高等职业教育相配套的教材建设也迅猛发展，各大出版社均推出一套或几套高等职业教育教材。相比之下，高等职业教育类的辅导丛书并不多见。而在实际教学过程中，学生们常常反映没有适合的参考书。为了辅导广大应用型本科类、高职专科类学生更好地学好相关课程，清华大学出版社策划、组织编写了《21 世纪高等职业教育主干课程辅导丛书》。

本丛书作为学生正规教材的辅导用书，对课程的各方面知识不作细致复述，而是突出基本理论、基本概念、基本方法，配以典型例题解析，有助于读者理解和掌握本课程的主要内容，并且每章都附有对应的同步练习题和解答(或给出答案与提示)，附录中还提供了几套试题及其参考答案。

1. 丛书书目

本套丛书首批出版 6 本，书目如下：

(1) C 语言知识点与典型例题解析
(2) 数据结构知识点与典型例题解析
(3) 操作系统知识点与典型例题解析
(4) 高等数学知识点与典型例题解析
(5) 线性代数知识点与典型例题解析
(6) 概率论与数理统计知识点与典型例题解析

2. 特色提示

(1) 本丛书针对应用型本科类、高职专科类课时少的实际情况，定位于：讲清课本基本知识点，教会学生分析典型题，帮助理解、巩固所学知识。

(2) 丛书编写以“知识点讲解、典型题分析”为主线贯穿，以“辅导与训练并重，习题与分析结合”为原则。

(3) 注重基础，答疑解惑。突出“双基”(基础知识、基本能力)，对重点、难点、易混淆和易忽略的内容进行诠释与剖析，加强培养读者基本功，帮助读者掌握该课程的核心知识。

(4) 以典型题分析带动能力培养。本丛书以典型题目分析为突破口，强化各知识点的运用，培养读者应用能力。精心选取的例题具有典型性和针对性。所有例题均给出了详尽的分析，大部分例题还给出了点评，便于读者掌握完整的解题思路，以达举一反三、触类旁通之功效。

(5) 丛书基本按照正规教材顺序安排章节。每节包括四个板块，即基本知识点、典型题分析、同步练习题、同步练习题参考答案。各板块内容安排如下：

- 基本知识点　列出本章的基本理论、基本概念、基本方法，便于学生记忆、复习。
- 典型题分析　精选出基本题与经典题进行解析，增强学生解题能力。
- 同步练习题　设计出一部分习题，便于学生自测与检查。
- 同步练习题参考解答　给出习题的解答(或答案)，便于考生复习与检查。

(6) 附录中提供一套期中、期末考试试卷以及专升本考试样卷，所有试卷均给出解答，方便读者自学使用。

3. 作者与读者

丛书聘请具有较高理论水平和丰富教学经验的一线骨干教师编写。他们长期从事相关课程的教学、命题和研究等工作，积累了丰富的经验，对相应课程有较深的体会与独到的见解，本丛书凝聚了他们多年的教学经验和心血。

本套丛书特别适合参加课程学习、考试的学生读者群阅读，同时可供参加专升本的考生参考使用。

4. 交流与致谢

读者的进步是我们的心愿。如果发现书中有任何疑惑之处，或有建议和意见，请与我们交流。联系信箱：Reader_Service@126.com。

在此，对丛书所选用的参考文献的著作者，及丛书所精选的习题、试题的命题老师表示真诚的感谢。感谢为本丛书出版提供帮助的各界人士。

丛书编委会

前　言

数据结构是计算机专业教学的核心课程之一，数据结构课程不仅为计算机语言进行课程设计提供了方法性的理论指导，还是其后续课程学习的重要基础。数据结构侧重于体系和思想上的训练，是程序设计的灵魂，而语言仅是工具，是手段。

数据结构主要研究的是数据的各种组织形式，以及建立在这些结构上的各种操作及其实现。由于其所研究的对象——数据元素及其之间的关系都是从现实生活中抽象出来的，在被组织成不同形式时，只研究其抽象出来的本质性的各种概念和关系，而忽略其本身所代表的实际背景，因此在学习的时候会觉得比较抽象，特别是对于自学数据结构的读者，更不容易真正掌握它。

鉴于数据结构课程在整个计算机专业教学体系中的重要地位，我们特别编写了本辅导书。该辅导书以数据结构主流教材为蓝本，对重点和难点内容进行了深入的分析和剖析，着重强调培养学生的动手能力，对相关的经典算法、重点算法及应用算法进行详细的解答，所有的源程序均上机通过，以便学生更好地掌握数据的存储结构的表示与运用。

本书共分 9 章，覆盖了数据结构课程的各部分内容，力求同时兼顾题目的广度和深度。每一章包括典型例题分析和练习题两个部分。典型例题分析和练习题是根据相应章节的重要内容的知识点进行精心挑选的，给出了比较详细的分析和解答，并对部分题目进行了点评的拓展分析，在对试题的部分题目进行点评中，诱导读者深入思考问题的本质，拓展读者的思路。

作为本书的作者，我们希望广大读者能够通过本书，掌握数据结构的知识和原理，提高运用这些知识解决实际问题的能力，为掌握程序设计和软件开发的方法奠定扎实的基础。

本书可作为相关高校数据结构课程的参考书，也可作为相关考试(自学考试；专升本、程序员 、计算机等级考试(三级)等)的辅导用书。

本书由杨明主编，吉根林、严云洋副主编，许勇主审。成书的过程中，资料的收集、整理工作由杨萍、汪志宏、石雪梅、田玉敏、张凌云、谢波、王国全、赵传申、何光明、陈智等共同完成，在此一并感谢。

由于本书中题目数量大，解答中难免会出现欠考虑的地方，敬请读者批评指正。

特别说明：配书源程序可以到www.wenyuan.com.cn下载。

编　者

目　录

第1章 绪 论

要点导读：

- 掌握数据结构的基本概念
- 了解数据结构课程的重要性
- 了解算法描述的基本方法
- 掌握算法的基本特征

1.1 引 言

用计算机解决任何问题都离不开程序设计。为了编制有效可行的程序，人们开始分析研究程序所处理的数据的特性及数据之间的关系，这就是“数据结构”这门学科形成和发展的背景。

数据结构主要研究非数值应用问题中数据之间的逻辑关系和对数据的操作，同时还研究如何将具有逻辑关系的数据按一定的存储方式存放在计算机内。分析数据之间的逻辑关系和确定数据在计算机内的存储结构是程序设计之前必须完成的两个任务。

用计算机解决现实问题的方法一般分为这样几个步骤：(1)分析现实问题并得到相应的数学模型；(2)设计出解决该数学模型的算法；(3)编程实现该算法从而得到问题的解。数据结构几乎体现在问题求解的各个步骤，尤其是步骤(2)。可见，数据结构在计算机课程中起着核心作用，学好数据结构具有十分重要的意义。

同时，数据结构既依赖于前续课程的学习，使得前续课程得到应用和巩固；也为后续课程的学习打下扎实的基础，尤其是“编译原理”及“操作系统”等课程。

【例1】简答题

1. “数据结构”课程是计算机专业的基础课、专业课，还是专业基础课？

2. 学习“数据结构”课程需要哪些课程作为它的基础(列举两门课程)？若没有这些知识，对学习“数据结构”课程可能会产生哪些影响？请举例说明(不超过100字)。

3. “数据结构”课程将为哪些课程的学习奠定必要的基础？请举例说明哪些课程(列举两门课程)用到了“数据结构”课程中的哪些知识(不超过100字)。

【分析】

此题主要考核“数据结构”课程的性质，并以此强调其在计算机科学中的重要性。

【解答】

1. “数据结构”课程是计算机专业的专业基础课。

2. 学习“数据结构”课程需要一些课程作为它的基础，如“C语言”与“离散数学”。若没有C语言或其他语言基础(如Pascal或C++)，则学生就难以理解描述数据结构及其算法的类C或类Pascal或类C++，更重要的是会造成学生上机环节的困难，影响该课程的学

习。同样，“离散数学”课程是“数据结构”课程的理论基础，其中集合、树及图等重要理论知识为“数据结构”课程的学习提供了重要的理论基础。

3. “数据结构”课程为“编译原理”、“数据库系统”和“操作系统”等课程的学习奠定了必要的基础。例如：“编译原理”的表达式求解要用到“数据结构”的栈知识，“编译原理”的符号表管理技术要用到“数据结构”的散列查找知识；“数据库系统”的存储要用到“数据结构”的B+树知识；“操作系统”的设备链表管理要用到“数据结构”的线性链表知识，“操作系统”的最短作业优先服务要用到“数据结构”的队列知识。

【点评】

深刻理解“数据结构”课程的性质和内容，就不难掌握和理解其与其他课程之间的关系。例如，数据结构算法的描述和实现需要借助一种高级语言，而目前比较流行的高级语言有C、C++、Java、Pascal等，因而C和Pascal可作为数据结构课程的前续课程。同理，“操作系统”、“编译原理”及“数据库系统原理”等课程中需要用到大量的数据结构知识，例如，“操作系统”中的作业队列、就绪进程队列及设备队列等；“编译原理”中借用栈来进行表达式求值等。

1.2 基本概念和术语

- 数据(Data)：数据是信息的载体，是指所有能输入到计算机中并被计算机程序处理的内容总称。计算机输入和处理的数据除数值外，还包括字符串、表格、图像和声音等，它们都属于数字编码范畴。
- 数据元素(Data Element)：数据元素是数据的基本单位。数据元素也称为元素、结点、顶点、记录。一个数据元素可以由若干个数据项(也可称为字段、域、属性)组成。数据项是具有独立含义的最小标识单位。
- 数据结构(Data Structure)：数据结构指的是数据之间的相互关系，即数据的组织形式。数据结构常指数据的逻辑结构，它分为线性结构(如栈、队列、串)和非线性结构(如数组、广义表、树和图等)两种。
- 数据的逻辑结构(Logical Structure)：数据的逻辑结构是指数据元素之间存在的一种或多种特定的关系，它从逻辑关系上描述数据，与数据的存储无关，是独立于计算机的。数据的逻辑结构可以看作是从具体问题抽象出来的数学模型。
- 数据的存储结构(Storage Structure)或数据的物理结构(Physical Structure)：数据元素及其关系在计算机存储器内的表示；数据的存储结构是逻辑结构用计算机语言的实现(亦称为映象)，它依赖于计算机语言。对机器语言而言，存储结构是具体的。一般只在高级语言的层次上讨论存储结构。
- 数据类型(Data Type)：这是和数据结构密切相关的一个概念，在用高级程序设计语言编写的程序中，每个变量、常量或表达式都对应一个确定的数据类型。数据类型可分为非结构的原子类型和结构类型两种。
- 数据的运算：即对数据施加的操作。数据的运算定义在数据的逻辑结构上，每种逻辑结构都有一个运算的集合。最常用的检索、插入、删除、更新、排序等运算

实际上只是在抽象的数据上所施加的一系列抽象的操作。所谓抽象的操作，是指我们只知道这些操作是“做什么”，而无须考虑“如何做”。只有确定了存储结构之后，才考虑如何具体实现这些运算。

- 算法(Algorithm): 指解决特定问题的一种方法或一种描述。

1.3 算法描述

算法是指解决问题的一种方法或过程描述。如果将问题看作函数，则算法能把输入转化为输出。解决一个问题可有多种算法，但一个给定的算法只能解决一个特定的问题。可见，算法不同于程序。算法应满足以下5个特性：

- 正确性 它必须解决具体的问题，完成所期望的功能，给出正确的输出。
- 确定性 算法执行的每一步和下一步必须确定，不能有二义性。
- 有限性 一个算法必须由有限步组成。算法必须可以终止，不能进入死循环。
- 输入 一个算法有零个或多个输入。
- 输出 一个算法有一个或多个输出。

根据待解决实际问题的不同，通过算法实现的基本操作也有差异，但数据结构上的主要基本操作有：

- 查找 寻找满足特定条件的数据元素所在的位置。
- 读取 读出指定位置上数据元素的内容。
- 插入 在指定位置上添加新的数据元素。
- 删除 删除指定位置上对应的数据元素。
- 更新 修改某个数据元素的值。

同时，根据操作的结果可将操作分为加工型操作和引用型操作两种。其中，加工型操作改变原逻辑结构的“值”，如插入、删除、更新、调整等，而引用型操作不改变原逻辑结构的“值”，如查找和读取。

算法的描述方法有很多，根据描述算法的语言不同，描述方法常分为以下4种。

- 框图算法描述：该方法具有直观、易懂等优点，但不够清晰简洁，不便于描述复杂的算法。
- 非形式算法描述：用中文语言与一些程序设计语言中的语句相结合来描述算法。
- 类C语言算法描述：也称伪语言算法，不能直接在计算机上运行。
- C语言编写的程序或函数：可在计算机上运行并获得结果的算法，也称为程序。

1.4 算法分析

为设计一个“好”的算法，要求算法设计考虑以下3个方面：

- 可读性 算法应易于阅读和理解，以便于调试、修改和扩充。
- 健壮性 无论正确的输入，还是不正确的输入，算法都能进行相应的处理。
- 高效率 即达到所需的时空性能。算法的时空性能是指该算法的时间性能(时间效

率)和空间性能(空间效率)。

衡量算法时间效率不是采用绝对的运行时间计算，而是指算法在某个问题“规模”上执行“基本操作”的次数。

1.5 同步练习题

一、选择题

1. (　　)不是算法的基本特征。

A. 正确性　　B. 长度有限

C. 在规定的时间内完成　　D. 确定性

2. 计算机中的算法指的是解决某一问题的有限运算序列，它必须具备输入、输出、(　　)等五个特征。

A. 可执行性、可移植性和可扩充性　　B. 可执行性、有穷性和确定性

C. 确定性、有穷性和稳定性　　D. 易读性、稳定性和确定性

二、填空题

1. 数据的逻辑结构被分为__(1)__、__(2)__、__(3)__和__(4)__4 种。
2. 在图形结构中，每个结点的前驱结点和后续结点数可以_____。
3. 算法的 5 个重要特性是__(1)__、__(2)__、__(3)__、__(4)__和__(5)__。
4. 一种抽象数据类型包括__(1)__和__(2)__两个部分。
5. 当问题的规模 n 趋向无穷大时，算法执行时间 T(n)的数量级被称为算法的_____。

三、简答题

1. 简述数据结构的逻辑结构和存储结构的区别与联系。它们如何影响算法的设计与实现?
2. 简述常用的数据结构有哪些。

四、算法题

假设 n 为 2 的乘幂，并且 n>2，试求下列算法的时间复杂度及变量 count 的值(以 n 的函数形式表示)。

```
Int Time (int n){
count=0;  x=2;
While (x<n/2) {
x * =2;count++;
        }
return(count)
       }//Time
```

1.6　同步练习题参考解答

一、选择题

1.

【分析】

主要考查算法的 5 个特征。算法应满足有穷性、确定性、可行性、输入和输出 5 个基本特性。

【解答】

长度有限并不是算法的特性之一，因而 B 是错误的。

2.

【解答】

类似题 1 的分析，选 B。

二、填空题

1.

【解答】

(1)集合　　(2)线性结构　　(3)树形结构　　(4)图形结构

2.

【解答】

任意多个

3.

【解答】

(1)有穷性　　(2)确定性　　(3)可行性　　(4)输入　　(5)输出

4.

【解答】

(1)数据　　(2)操作

5.

【解答】

复杂度。

三、简答题

1.

【解答】

若用结点表示某个数据元素，则结点与结点之间的逻辑关系就称为数据的逻辑结构。数据在计算机中的存储表示称为数据的存储结构。可见，数据的逻辑结构是反映数据之间的固有联系，而数据的存储结构是数据在计算机中的存储表示。尽管因采用的存储结构不同，逻辑上相邻的结点，其物理地址未必相邻，但可通过结点的内部信息，找到其相邻的结点，从而保留了逻辑结构的特点。

由于采用的存储结构不同，该数据结构上的操作灵活性、算法复杂度等区别较大。

2.

【解答】

常用的数据结构或逻辑结构有集合、线性结构、树形结构和图状结 4 种。

四、算法题

【分析】

该算法的主要操作步骤为乘法，只要算法中的乘法执行次数即可。

【解答】

若设乘法的执行次数为 f(n)，则 $f(n)=\lceil lb(n/2)\rceil-2$ (n>4),f(n)=0(n<=4)。count 的值就是 f(n)。可见，该题算法的时间复杂度为 $O(lbn)$。

【点评】

从该题可知，对任一算法，只要得到与该算法对应问题规模的函数，便可求得该算法的时间复杂度，而算法对应问题规模的函数通常与循环次数有关。

第2章 线 性 表

要点导读：

- 了解线性表的概念
- 掌握线性表的基本操作
- 理解线性表的顺序存储结构及其上基本操作的实现
- 能灵活运用线性表解决实际问题

2.1 线性表的定义及逻辑结构

线性表的基本特征清晰地反应了线性表数据结构的特点，其基本特征为在一个非空线性结构中，有且只有一个称为第一个的元素；有且只有一个称为最后一个的元素；第一个元素无前驱，最后一个元素无后继；其余每个元素均有惟一前驱和惟一后继。

从数学的角度来看，Linear_list=(D, R)

其中，D 是数据元素的集合：$D=\{ a_i | a_i \in D_0, i=1, 2, \cdots, n, n>=0\}$；

R 是数据元素间关系的集合：R={N}；

$N=\{<a_{i-1}, a_i> | a_{i-1}, a_i \in D_0, i=1, 2, \ldots, n, n>=0\}$；

D_0 是具有某种性质的数据元素的集合。

【实例 1】英文字母表(A，B，…，Z)是线性表，表中每个字母是一个数据元素(结点)。该线性表的长度 n=26，按照上述数学描述，$D_0=\{$ A，B，…，Z $\}$，R={<A，B>，<B，C>，<C，D>，…，<Y，Z>}，线性表 Linear_list=(D，R)。

【实例 2】若 $D_0=\{$ A，B，C $\}$，R={<A，B>，<A，C>}，则 Linear_list=(D，R)就不是线性表，因为数据元素 A 有两个后继。

2.2 线性表的基本操作

线性表的基本操作包括以下几个方面。

(1) Initiate(L) 构造一个空的线性表 L，即表的初始化。

(2) Length(L) 求线性表 L 中的结点个数，即求表长。

(3) Get(L,i) 取线性表 L 中的第 i 个数据元素，这里要求 $1 \leqslant i \leqslant$ ListLength(L)。

(4) Locate(L,x) 定位函数。在 L 中查找值为 x 的数据元素，并返回该数据元素在 L 中的位置。若 L 中有多个数据元素的值和 x 相同，则返回首次找到的数据元素位置；若 L 中没有数据元素的值为 x ，则返回一个特殊值表示查找失败。

(5) Insert(L,x,i) 在线性表 L 的第 i 个位置上插入一个值为 x 的新数据元素，使得原编号为 i，i+1，…，n 的数据元素变为编号为 i+1，i+2，…，n+1 的数据元素，其中 $1 \leqslant i \leqslant n+1$，而 n 是原表 L 的长度。插入后，表 L 的长度加 1。要特别注意插入的位置 i 是否满足条件。

(6) Delete(L,i) 删除线性表 L 的第 i 个结点，使得原编号为 i+1，i+2，…，n 的结点变成编号为 i，i+1，…，n-1 的结点。这里 1≤i≤n，而 n 是原表 L 的长度。删除后表 L 的长度减 1。

(7) Empty(L) 判空表函数。L 为空，返回值为真；否则返回值为假。

(8) Clear(L) 将 L 清空。

2.3 线性表的顺序存储结构

线性表本身的定义是个逻辑的概念，要在计算机上实现线性表的基本操作就必须给定其存储结构。线性表的顺序存储是计算机中最简单、最常用的一种存储方式，即用一组地址连续的存储单元依次存放线性表中的元素，该结构的特点是逻辑相邻的数据元素在物理上也相邻。用顺序存储方法存储的线性表简称为顺序表(Sequential List)。

设线性表中每个数据元素占用 c 个存储单元，其中第一个单元的存储地址是该数据元素的存储地址，并设表中开始数据元素 a_1 的存储地址(简称为基地址)是 $LOC(a_1)$，那么数据元素 a_i 的存储地址 $LOC(a_i)$可通过下式计算：

$$LOC(a_i)= LOC(a_1)+(i-1)*c\ ,\quad 其中，1\leqslant i\leqslant n$$

顺序表类型定义如下：

```
#define ListSize 100              /* 线性表空间的大小*/
typedef struct {
 Elemtype data[ListSize+1];       /*存放线性表数据元素的数组 data*/
 int length;                      /*记录线性表的当前长度*/
}SeqList;
```

对上述定义顺序表类型 SeqList，要特别注意的是：

(1)数组的下标从 0 开始，而不是从 1 开始。为了和线性表的逻辑定义一致，下标为 0 的数组元素不用，从下标 1 的开始用。

(2)线性表改变时，要及时修改其中 length 成员变量的值。

2.4 基本操作在顺序表上的实现

2.4.1 基本知识点

由于 C 语言中数组的下标是从 0 开始的，所以，在逻辑上所指的“第 k 个位置”实际上对应的是顺序表的“第 k-1 个位置”。这里仅介绍在顺序表上线性表的插入、删除和定位函数。

(1) 插入函数

```
insert(SeqList *L, elemtype  x, int i) /*该算法在线性表 L 的第 i 个位置插入元素
                                        x*/
 { if ((i<1)||(i>L->length+1))          /*插入位置非法*/
    printf("error");
   else
```

```
for(j= L->length;j>=i;j--)
 L->data[j+1]=L->data[j];
    L->data[i]=x; L->length++;
  }
```

(2) 删除函数

```
delete (SeqList *L, int i)   /*该算法在线性表 L 的第 i 个位置删除元素*/
  { if ((i>L->length)||(i<1))  printf("error");
else
 { for (j=i;j<=L->length-1;j++)
       L->data[j]=L->data[j+1];
     L->length--;
    }
  }
```

(3) 定位函数

```
int Locate(SeqList L,elemtype x) /*在线性表 L 中查找数据元素 x*/
   {
 int k=0;
 while(k<=L.length&&L.data[k]!=x) k++;
 if(k<=L->length) return k;
  else retrun 0;
}
```

插入和删除元素算法的时间复杂度分析如下：

(1) 插入算法的时间复杂度

$$\sum_{i=1}^{n+1}(pi*ci)$$

/*pi 是在第 i 个元素前插入元素的概率 */

/* ci 是在第 i 个元素前插入元素时元素移动次数*/

$$=1/(n+1)\sum_{i=1}^{n+1}(n-i+1)$$

$$=\frac{n}{2}$$

(2) 删除算法的时间复杂度

$$\sum_{i=1}^{n}(pi*ci)$$

/*pi 是在第 i 个元素前删除元素的概率 */

/* ci 是在第 i 个元素前删除元素时元素移动次数*/

$$=1/n*\sum_{i=1}^{n}(n-i)$$

$$=\frac{n-1}{2}$$

可见，插入和删除算法的时间复杂度均为 O(n)。