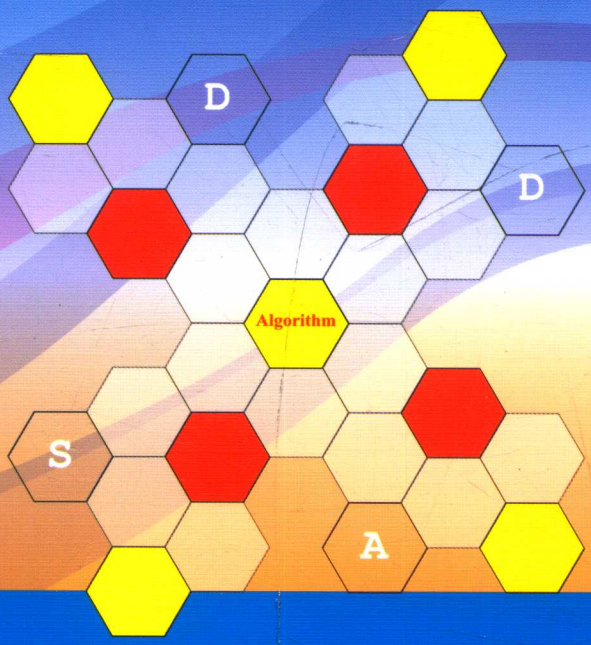
高等学校计算机专业规划教材

面向算法设计的数据结构

（C++语言版）



谢　飓　编著

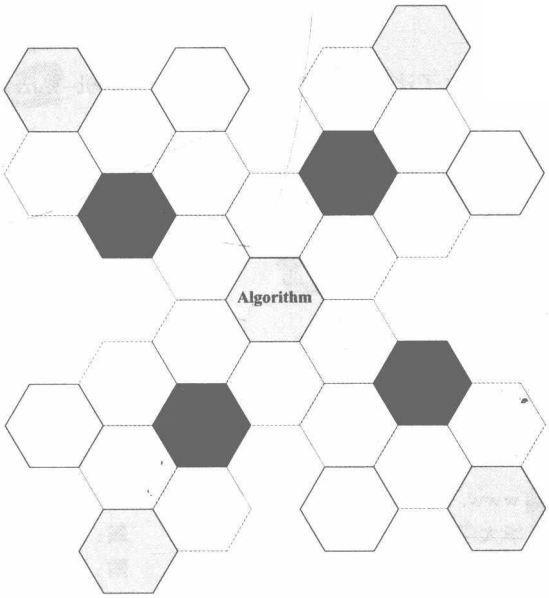
清华大学出版社

高等学校计算机专业规划教材

面向算法设计的数据结构

（C++语言版）

谢　勰　编著



清华大学出版社
北京

内 容 简 介

本书基于抽象数据类型的观点讲解数据结构，力图让读者学会以“积木式”组件方案快速、便捷、高效地构建程序. 数据结构要为算法服务，因此本书以算法分析为导向，以算法效率为准绳，着墨于抽象数据类型的选择、使用和组合，从而实现提升算法性能的终极目标. 全书采用 C++语言描述程序，并尽量与 C++ 11 标准靠拢，力求紧跟程序设计语言的时代脉搏. 本书特色在于以标准模板库（STL）高效地编写 C++ 程序代码，并特别论及了各种容器的算法性能优劣，从而让读者能够更好地使用 STL 容器.

本书可作为高等院校计算机科学与技术等本科专业的数据结构课程教材，也可供相关专业的工程技术人员参考.

图书在版编目(CIP)数据

面向算法设计的数据结构：C++语言版/谢勰编著. --北京：清华大学出版社，2015
高等学校计算机专业规划教材
ISBN 978-7-302-41152-9

Ⅰ.①面… Ⅱ.①谢… Ⅲ.①电子计算机-算法设计 ②电子计算机-算法分析 ③C语言-程序设计 Ⅳ.①TP301.6 ②TP312

中国版本图书馆 CIP 数据核字(2015)第 183515 号

责任编辑：龙启铭
封面设计：何凤霞
责任校对：焦丽丽
责任印制：何　芊

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn,010-62795954

印 装 者：北京鑫海金澳胶印有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：18.25　　**字　　数**：445 千字
版　　次：2015 年 12 月第 1 版　　**印　　次**：2015 年 12 月第1 次印刷
印　　数：1～2000
定　　价：39.00 元

产品编号：037766-01

面向算法设计
的数据结构
C十十语言版

算法运行情况, 更能提高算法设计水平. 例如初学者按照此角度可时时提醒自己要按照"机械式"方式给出步骤, 而专业人士则能在纷繁复杂的数据和算法步骤中找到规律进而优化. 通俗地说, 要把自己想象成计算机, 这样身临其境方能参透奥秘.

在许多教材中, 算法只是观念上的实现, 从而导致教材自身不注重具体代码的编写, 即便给出代码也未精化. 我们希望能给出清晰高效且具备良好代码风格的C++语言实现, 并且尽量兼顾学术前沿和工程实践. 为此, 我们创建了DSAD项目(`https://github.com/xiexiexx/DSAD`), 本书代码可在其中下载, 并将不断维护更新, 该项目还会定期发布全书的勘误表.

组织

教材内容组织安排上遵循以抽象数据类型为核心的思想, 突出如何以抽象数据类型搭建算法和应用程序. 为此, 本书中大部分章节标有分类指示:

- [使用] 主要讲解STL中容器的用法, 它解决抽象数据类型如何用的问题.
- [技巧] 深入剖析算法与编程的技巧, 它主要关注效率的提升.
- [实例] 针对具体案例给出完整的解决方案, 它展示了抽象数据类型的典型应用场景.

说明

- 本书向C++11标准靠拢, 因此请读者尽量使用新版本的编译器. 不过考虑到编程习惯和读者的接受能力, 我们没有完全使用C++11, 例如基于范围的`for`循环和空指针`nullptr`等新特性.
- 我们将常用的头文件和名字包含于`book.h`, 并作为书末附录, 读者包含此文件即可执行书中所涉及的STL库. 此外, `book.h`还包含了本书所实现的头文件, 这样使用本书的代码会更加方便.
- 在一般情况下, 本书中log一般指代$\log_2$, 即我们主要讨论以2为底的对数.
- 不同字体代表不同对象: 例如x代表一个程序中使用的变量, 而x表示一个数学公式中的符号.
- 本书仅讲解一些基本的排序和查找内容, 并分散于不同章节, 而更深入的相关内容留待算法课程中再续.
- 书中加星号的章节和习题为选学与选做内容, 读者可根据需要取舍.

致谢

在本书的写作过程中, 得到了诸多人士的大力支持, 感谢: 朱寅(Citadel)对全书内容深入有益的讨论, 李树钧(英国Surrey大学)为本书的LaTeX排版提出了很好的建议, 张志强(中信证券)长期与作者交流探讨算法问题, 陈硕(Google)分享了许多他对于C++语言的精辟见解, 且为DSAD项目奉献良多, 王辉(西安邮电大学)认真细致地审读了全书. 感谢本书编辑

龙启铭先生对作者的耐心包容. 感谢我的父亲为本书题字. 特别还要感谢试用本书的各位同学, 正是他们促进了这本书的不断完善.

由于作者水平所限, 恳请大方之家对书中所出现的问题提出指正! 读者朋友可将您的意见和建议发送至DSAD2015@163.com.

谢 勰

2015年6月于西安邮电大学

目录

第1章

算法

计算的目的是什么?

在回答这个问题之前, 先要明确计算(computation)这个词早已突破了传统意义上的运算操作, 它具有更宽泛的意义. 简单地说, 计算是解决人类以手工方式难以解决的问题. 许多问题如果使用人来解决, 则会需要大量的人力与时间,[1] 从人类的效率特点而言, 可能这些问题是"无法解决"的. 例如, 在互联网上收集信息以便搜索使用, 只能利用网络爬虫来完成这种极其繁杂的工作. 当然, 计算并不意味着笨拙的苦力工作. 许多我们认为是只能依靠人类智慧才能解决的问题, 如今通过高质量的计算也可以达到目的, 深蓝(Deep Blue)战胜卡斯帕罗夫(Garry Kasparov)就是一个有力的佐证. 事实上, 在计算的过程中, 一个至关重要的概念就是算法(algorithm). 从某种意义上说, 算法是在计算中实现人类智慧的关键枢纽.

1.1 概述

何谓算法?

在程序设计课程中, 我们已经粗略接触过算法, 但未仔细考虑这个词的意义. [Knu97a]比喻算法有点像烹饪法(recipe), 也像方法(method), 还像过程(procedure), 更像技术(technique). Merriam-Webster词典为我们浅显易懂地解释了算法[2]这个词:

> A procedure for solving a mathematical problem (as of finding the greatest common divisor) in a finite number of steps that frequently involves repetition of an operation; broadly: a step-by-step procedure for solving a problem or accomplishing some end especially by a computer.

更为专业的"定义"则来自[Knu97a], 它将算法描述为:

> Besides merely being a finite set of rules that gives a sequence of operations for solving a specific type of problem, an algorithm has five important features: 1) Finiteness. .... 2) Definiteness. .... 3) Input. .... 4) Output. .... 5) Effectiveness. ....

上述看似简单却相当精确的描述非常值得我们深入思考. 具体地说, 算法必须具备5个基本特性:

- 有穷性(Finiteness): 算法必须在有限步内结束.
- 确定性(Definiteness): 算法的所有步骤都必须精确定义, 不得模棱两可.
- 输入(Input): 算法通常得有输入数据, 不过输入可以为空.

[1] 这种花费可以用人年这个概念描述, 粗略地说就是人在一年内能完成的工作量.

[2] http://www.merriam-webster.com/dictionary/algorithm.

- 输出(Output): 算法肯定会得到一个或多个输出数据. 输出和输入是有关系的, 而这种关系的确认通常会涉及算法是否"正确"地实现了设计者的意图.
- 能行性(Effectiveness): 算法的所有运算对于执行者而言都应是"基本"运算. "基本"这个词意味着原则上[1]人可以用纸笔精确复原并完成算法的执行过程.

需要注意, 这些特性组合起来会得到更多的要求, 例如输入和输出的数据量都必须是有限的, 否则无法满足有穷性.

算法是有穷、确定且能行的过程, 它处理所输入的数据(也可能是空数据)而获得输出结果.

例 1.1 插入排序是一个简单的排序方法, 以输入数据为2，5，3，1，4为例, 从算法的角度分析插入排序.

解: 我们按照算法的5个特性来分析. 本例中输入是2，5，3，1，4. 插入排序每步依次将数据取出, 再按照一定的规则插入到已排序的序列中, 容易验证该方法满足确定性. 可以在纸上写出过程:

```
第1步：2                    // 取出2插入.
第2步：2，5                 // 取出5插入.
第3步：2，3，5              // 取出3插入.
第4步：1，2，3，5           // 取出1插入.
第5步：1，2，3，4，5        // 取出4插入.
```

从上述过程可以看出插入排序满足有穷性和能行性. 最后, 插入排序在本例中的输出是1，2，3，4，5.

事实上, 可以证明插入排序满足算法的5个特性, 因此它可称为一个算法. 当然, 有穷性需要严格的证明, 不妨留作练习. 从更深入的角度看, 有穷性的判定属于停机问题(halting problem).

顺便提及, 插入排序来源于现实生活, 图1.1演示了对扑克牌进行插入排序的过程. 事实上, 许多算法的设计都需要从实际中寻找灵感. ⋆

算法体现着一种思维. 对于海量数据, 只能使用算法思维, 不能依照人的方式随意处理. 首先, 人类在处理大量重复性和规律性的操作步骤时非常容易出错, 这点远不如机器. 其次, 人的许多手工处理方式是无规则而且杂乱的, 这种随意的处理在数据量较大时根本无法实施. 最后也是最重要的是, 机器只能按照规则行事. 因此, 算法是处理困难问题的一种必然选择. 不夸张地说, 算法是机器的世界观.

如今, 算法的应用越发广泛, 在计算机领域的各个方面都有它的身影, 例如人工智能、数据库、计算机网络、操作系统、计算机图形学等, 不仅如此, 算法在其他学科领域中也发挥着巨大的作用, 例如信号处理、模式识别、集成电路设计. 更奇妙的是, 算法和一些传

[1] 许多算法所需要的人、时间和空间等元素是现实中无法实现的.

统的学科交融形成了新兴的研究领域, 其中最引人注目的就是计算生物学, 它给算法和生物学研究都带来了众多新热点.

正因如此, 我们需要深刻地掌握算法思维, 努力设计出优秀的算法, 这样才能更好地服务于实际. 从这个角度讲, 算法还应该具备更多的优良特性, 例如算法应该高效而且具有鲁棒性, 还应该形式优雅且使用简易方便. 要设计这样的算法, 必须依赖于我们的深入学习和不断实践.

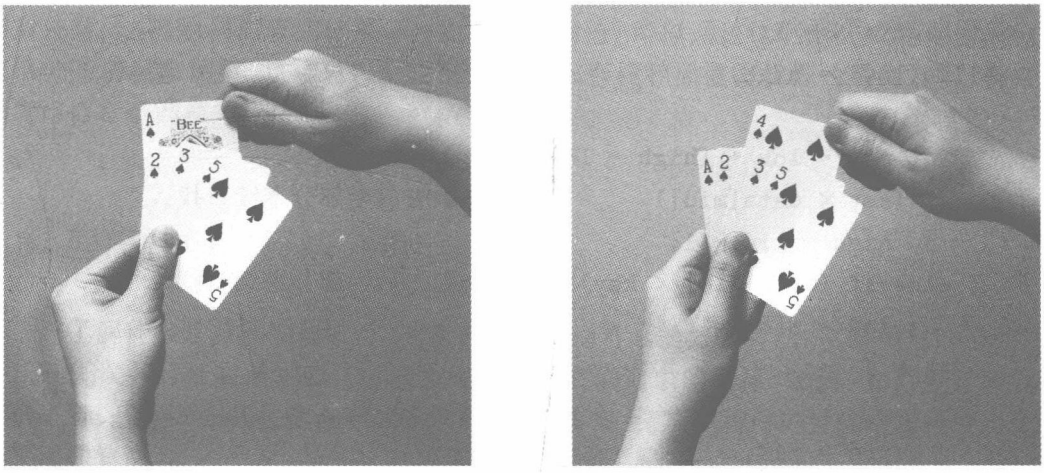


图 1.1 对扑克牌进行插入排序

1.2 [实例] 二分查找

本节的问题是在一个已经过整理的历史数据集中查找, 说得更准确些, 是在有序排列的数组data中查找某个值key. 很容易找到较好的实例, 例如在字典中查找某个词,[1] 一个简单的方案是翻到字典的中间位置, 根据与当前位置单词的比较来决定下次应该选择哪一半作为新的待查空间, 这种方法一般称为二分查找(binary search).

此类查找问题在数学分析中的一般提法是: 在函数$f(x)$的单调区间$[a,b]$中寻找方程$f(x)=0$的根. 此类求根问题解决起来不是很难, 已有很多成熟的方法, 二分查找就是其中之一. 由于对实值的操作(例如取中点和端点处理)较为简单, 因此程序的编写也不麻烦, 只是对于数据精度要求稍多一些. 但是, 在离散的数据集中进行二分查找就需要考虑更多的问题.

我们先看看二分查找的程序实现应该是什么样子:

◁ 二分查找 ▷

```
template <typename T>
void BinarySearch(const T& key, const T data[], int N,
```

[1] 英语词典的查找有更好的方法和数据结构, 有兴趣的读者可查阅相关资料.

```
    bool& found, int& pos)
{
    // 输入: key - 待查值; data - 有序数据集(从小到大排列); N - 数据集长度.
    // 输出: found - 标记key是否找到; pos - 所找到的位置.
    found = false;
    pos = N;
    int low = 0;
    int high = N - 1;
    while (low <= high)
    {
        int mid = low + (high - low) / 2;
        if (key < data[mid])          // 小则去前半部分继续查找.
            high = mid - 1;
        else if (data[mid] < key)     // 大则去后半部分继续查找.
            low = mid + 1;
        else
        {
            pos = mid;
            found = true;
            break;                    // 找到则跳出循环.
        }
    }
}
```

► 二分查找 ◄

读者可以自行编写或者更改我们的二分查找程序, 再对比分析是否会存在错误. 特别要指出, 算法的实现相当重要, 例如它得用良好结构化的程序语言完成, 逻辑要清晰, 所用语法不能太冷僻等. 这些都应从二分查找的程序实现中仔细体会.

为了判断二分查找程序是否是一个算法, 可从算法的5个特性分析:

- 该程序在循环体内的关键在于三路分支的判断: 如果小于则转向前一半, 反之再判断如果大于则转向后一半, 如果都不满足最后必然是相等的情况, 此时应跳出循环. 这三类情况下都趋向于缩短当前的区间或者跳出循环, 可保证循环的终结. 因此, 二分查找程序满足有穷性.
- 该程序具备输入, 并规定了输入数据的形态, 以接口的形式对外宣布.
- 该程序能输出数据, 并且数据形式清楚明白地反映了问题的答案. 可以证明: 如果key可查到, found为true, 且返回的位置pos对应的值为key; 反之found为false, 并返回一个输入中不存在的pos位置(这里定为N).
- 该程序已经用C++程序设计语言实现且编译无误, 因此是确定的.
- 该程序所使用的语句都在C++程序设计语言所能完成的范围之内, 因而是能行的.

综上可知, 二分查找程序是一个算法. 事实上, 上述分析也加深了我们对算法的认识. 此外, 需要注意程序设计语言本身在判定程序是否满足算法特性中的功用.

顺便提及, 单纯从结构化程序设计的观点来看, 程序中使用了`break`语句. 尽管可利用`high = low - 1;`或`low = high + 1;`来跳出循环, 但考虑到这样不利于程序的直观性, 而且如果修改程序也容易带来不便. 不过从理论角度看, 这样的方式将算法的终结条件统一改成了空区间, 因而也别具风格.

此外, 查字典的方法有助于我们提出更好的查找思路. 比如我们根据字的读音判断出相对位置, 这提示我们查找位置不一定要取中间值, 可以通过比例形式或者根据具体数据取值分布情况来调整. 这样可以衍生出Fibonacci查找和插值查找等一系列方案, 有兴趣的读者可自行查阅相关资料.

1.3 程序性能与算法分析

在了解什么是算法的基础上, 非常有必要知道什么是好的算法. 从最简单的层面看, 一个正确无误并且能高效完成任务的算法肯定是我们追求的目标. 显然, 如果要对所设计的算法进行评估, 那么其关键在于算法性能指标. 事实上, 应用程序的性能很大程度上依赖于所采用的算法, 从这个意义上讲, 算法是程序的灵魂.

运行时间和占用空间是算法性能最关键的指标. 可以考察这些指标的实际值, 比如可利用计时工具(本书附录中提供了一个`xTimer`类)来考察运行时间, 又如操作系统中的相关工具[1]可以查看占用空间. 从效率观点看, 高效的算法意味其执行过程消耗的机器资源较少, 而计算机系统必要的构成是CPU和内存,[2] 那么肯定得要求算法的执行过程应尽量少占用处理机时间和内存.

不过, 这些具体且"实际"的指标值却是不"实用"的性能指标. 一方面, 不同的计算机由于性能各异造成指标值不同; 另一方面, 这些指标值随着输入数据不同而相差较大. 显然, 可以在某些条件上给出具体的指标值, 但这无法适用于所有情况. 鉴于此, 理论分析工具的呼唤是必然的, 而如今此方面的研究已经成长为*算法分析*(Algorithm Analysis)这个专门的领域.

算法分析一般是理论上的, 它可以在算法运行之前进行预判, 也可以在算法运行之后进行总结改进. 但算法分析代替不了实际的性能测试, 必须在真正的计算机上利用实际的编译器得到可执行的代码运行, 根据实测效果进行优化并重新给出更精准的分析, 这样才能达到最佳的效果.

对于算法效率, 可以引入两个指标分析, 它们分别从时间和空间上考虑:

- 时间复杂度(Time complexity). 算法完全运行所需运算时间.
- 空间复杂度(Space complexity). 算法完全运行所需存储空间.

这两个指标衡量了算法的时空效率.

---

[1] Windows操作系统下可运行perfmon.msc, Mac OS X中可使用Activity Monitor.

[2] 很多时候还需要外存的辅助, 现代意义上的算法也得考虑外存的重要性. 此外, GPU的作用也越来越大.