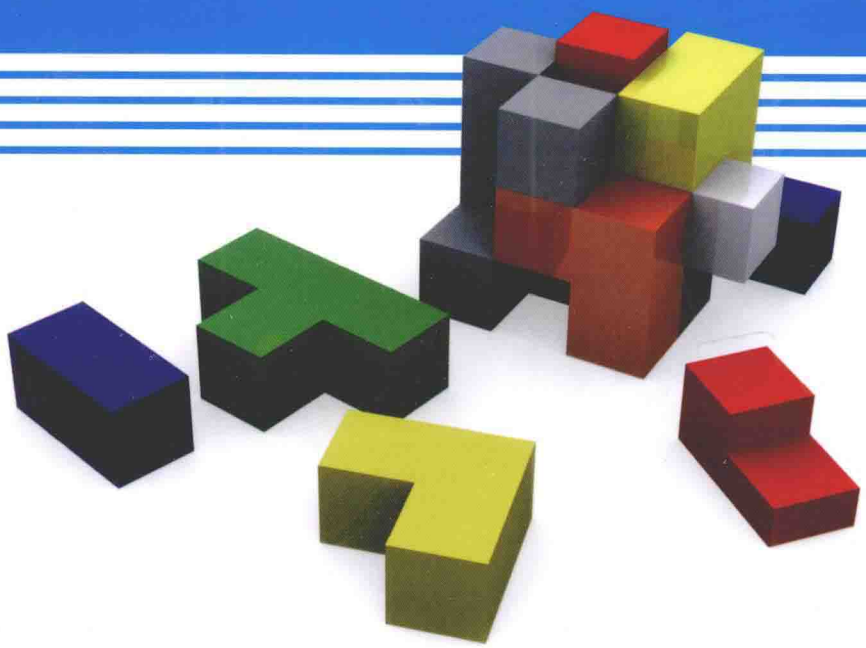
高等学校数据结构课程系列教材

# 算法设计与分析

李春葆　主编



清华大学出版社

高等学校数据结构课程系列教材

# 算法设计与分析

李春葆　主编
陈良臣　喻丹丹　曾平　编著

清华大学出版社
北京

## 内 容 简 介

本书系统地介绍了各种常用的算法设计策略，包括穷举法、分治法、贪心法、动态规划法、回溯法、分枝限界法等，并详细讨论了各种图搜索算法和计算几何设计算法。

全书既注重原理又注重实践，配有大量图表、上机实验题和练习题，内容丰富，概念讲解清楚，表达严谨，逻辑性强，语言精练，可读性好。

本书既便于教师课堂讲授，又便于自学者阅读。本书可作为高等院校算法设计与分析课程的教材，也可供 ACM 和各类程序设计竞赛者参考。



**图书在版编目(CIP)数据**

算法设计与分析/李春葆主编. --北京：清华大学出版社，2015
高等学校数据结构课程系列教材
ISBN 978-7-302-38113-6

Ⅰ. ①算…　Ⅱ. ①李…　Ⅲ. ①电子计算机－算法设计 ②电子计算机－算法分析　Ⅳ. ①TP301.6

中国版本图书馆 CIP 数据核字(2014)第 224393 号

**责任编辑**：魏江江　王冰飞
**封面设计**：杨　兮
**责任校对**：时翠兰
**责任印制**：杨　艳

**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 刷 者**：北京富博印刷有限公司
**装 订 者**：北京市密云县京文制本装订厂
**经　　销**：全国新华书店
**开　　本**：185mm×260mm　　**印　　张**：20.75　　**字　　数**：508 千字
**版　　次**：2015 年 5 月第 1 版　　**印　　次**：2015 年 5 月第1 次印刷
**印　　数**：1～2000
**定　　价**：39.50 元

产品编号：061559-01

FOREWORD

# 前言

算法在计算科学中扮演着重要角色。算法设计是计算机科学与技术专业的专业必修课，其目标是培养学生分析问题和解决问题的能力，使学生掌握算法设计的基本技巧和方法，熟悉算法分析的基本技术，并能熟练运用一些常用算法，解决一些较综合的问题。

在学习本课程之前，学生已经学习了基本的数据结构知识，能熟练运用一门或多门编程语言，并具备了一定的编程经验。如何利用这些已学过的知识，对不同的实际问题设计出有效的算法，正是本课程所要达到的目的。

本课程特点是“问题模型化，求解算法化，设计最优化”，在掌握必要的算法设计技术和编程技巧的基础上，能够在实际工作中根据具体问题设计和优化算法。本书是针对这一特点并结合本课程组的教学经验编写的。

全书由 10 章构成，各章内容如下：

第 1 章 概论，介绍算法的概念、算法分析方法和基本的计算复杂性理论。

第 2 章 递归算法设计技术，介绍递归的概念、递推式的计算、递归算法设计方法和相关示例、递归算法到非递归算法的转化。

第 3 章 穷举法，介绍穷举法的特点、穷举法的基本应用示例和递归在穷举法中的应用示例。

第 4 章 分治法，介绍分治法的策略和求解过程，讨论采用分治法求解排序问题、查找问题、最大连续子序列和问题、大整数乘法问题和矩阵乘法问题的典型算法，并简要介绍了并行计算的概念。

第 5 章 贪心法，介绍贪心法的策略、求解过程和贪心法求解问题应具有的性质，讨论采用贪心法求解区间问题、背包问题、多机调度问题、哈夫曼编码和磁盘排序问题的典型算法。

第 6 章 动态规划，介绍动态规划的原理和求解步骤，讨论采用动态规划法求解整数拆分问题、最长公共子序列问题、0/1 背包问题、最大连续子序列和问题和资源分配问题的典型算法。

第 7 章 回溯法，介绍解空间概念和回溯法算法框架，讨论采用回溯法求解 0/1 背包问题、子集和问题、排列和组合问题、迷宫问题、$n$ 皇后问题的典型

算法。

第 8 章 分枝限界法，介绍分枝限界法的特点和设计思想，讨论采用队列式分枝限界法和采用优先队列式分枝限界法求解 0/1 背包问题的典型算法。

第 9 章 图搜索算法设计，介绍图的存储表示、图的两种基本的搜索算法，利用 STL 设计算法的基本知识，讨论了构造图最小生成树的两种算法、产生图最短路径的 3 种算法，并采用 5 种算法策略求解旅行推销员问题(TSP 问题)以及求多段图的关键路径等典型算法；最后介绍网络流的相关概念以及求最大流和最小费用最大流的算法。

第 10 章 几何计算，介绍几何计算中常用的矢量运算以及求解凸包问题、最近点对问题和最远点对问题的典型算法。

另有 3 个附录，附录 A 给出部分练习题的参考答案，附录 B 给出所有上机实验题的参考程序，附录 C 给出书中主要算法实现程序的清单。

本书的特点是内容丰富、由浅入深、循序渐进，在各章中，首先介绍一种算法设计策略的基本思想，然后从解决实际问题入手，由易到难地描述几个经典示例，使读者既能学到一些常用求解问题的算法，又能通过对算法策略的反复应用，掌握其核心思想，以便收到融会贯通之效。同时本书特别注重同一个问题的多种解法以及不同算法的比较，使读者更容易体会到每一种算法策略的设计特点和各自的优缺点。

书中绝大多数算法在 VC++6.0 中调试通过，本书的教学 PPT 和所有源程序可以从清华大学出版社网站免费下载。

本书的编写工作得到湖北省教育厅和武汉大学教学研究项目《计算机科学与技术专业课程体系改革》的大力支持，清华大学出版社魏江江主任也全力支持本书的编写工作，作者在此一并表示衷心感谢。

本书是课程组全体教师多年教学经验的总结和体现，尽管作者不遗余力，由于水平所限，仍存在错误和不足之处，敬请教师和同学们批评指正，欢迎读者通过 licb1964@126.com 邮箱跟作者联系，在此表示万分的感谢。

编　者

2015 年 3 月

CONTENTS

# 目录

CHAPTER 1

# 概　论　第1章

算法是程序的灵魂，一个程序应包括对数据的描述（数据结构）和对操作的描述（算法）两个方面的内容，所以著名计算机科学家沃思提出一个公式，即数据结构＋算法＝程序。同一问题可能有多种求解算法，通过算法时间复杂度和空间复杂度分析判定算法的好坏。本章讨论算法设计和分析的相关概念，并简要介绍计算复杂性理论。

## 1.1　算法的概念

### 1.1.1　什么是算法

算法是求解问题的一系列计算步骤，用来将输入数据转换成输出结果，如图 1.1 所示。如果一个算法对其每一个输入实例，都能输出正确的结果并停止，则称它是正确的。一个正确的算法解决了给定的求解问题，不正确的算法对于某些输入来说，可能根本不会停止，或者停止时给出的不是预期的结果。

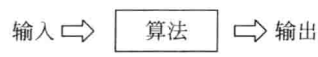



图 1.1　算法的概念

算法设计应满足以下几条目标。

（1）正确性：要求算法能够正确地执行预先规定的功能和性能要求。这是最重要的也是最基本的标准。

（2）可使用性：要求算法能够很方便地使用。这个特性也叫作用户友好性。

（3）可读性：算法应该易于人的理解，也就是可读性好。为了达到这个要求，算法的逻辑必须是清晰的、简单的和结构化的。

（4）健壮性：要求算法具有很好的容错性，即提供异常处理，能够对不合理的数据进行检查。不经常出现异常中断或死机现象。

(5) 高效率与低存储量需求：通常算法的效率主要指的是算法的执行时间。对于同一个问题如果有多种算法可以求解，执行时间短的算法效率高。算法存储量指的是算法执行过程中所需的最大存储空间。效率和低存储量这两者都与问题的规模有关。

**【例 1.1】** 以下算法用于在带头结点的单链表 h 中查找第 1 个值为 $x$ 的结点，找到后返回其逻辑序号(从 1 计起)，否则返回 0。分析该算法存在的问题。

```
#include <stdio.h>
typedef struct node
{   int data;
    struct node *next;
} LNode;                          //单链表结点类型定义
int findx(LNode *h;int x)
{   LNode *p=h->next;
    int i=0;
    while (p->data!=x)
    {   i++;
        p=p->next;
    }
    return i;
}
```

**解**：当单链表中首结点值为 $x$ 时，该算法返回 0，此时应该返回逻辑序号 1。另外当单链表中不存在值为 $x$ 的结点时，该算法执行出错，因为 p 为 NULL 时仍执行 p=p->next。所以该算法不满足正确性和健壮性。应改为：

```
int findx(LNode *h;int x)
{   LNode *p=h->next;          //p 初始时指向首结点
    int i=1;
    while (p!=NULL && p->data!=x)
    {   i++;
        p=p->next;
    }
    if (p==NULL)               //没找到值为 x 的结点返回 0
        return 0;
    else                       //找到值为 x 的结点返回其逻辑序号 i
        return i;
}
```

算法具有以下 5 个重要特征。

(1) 有限性：一个算法必须总是(对任何合法的输入值)在执行有限步之后结束，且每一步都可在有限时间内完成。

(2) 确定性：算法中每一条指令必须有确切的含义，不会产生二义性。

(3) 可行性：算法中每一条运算都必须是足够基本的，就是说它们原则上都能精确地执行，甚至人们仅用笔和纸做有限次运算就能完成。

(4) 输入性：一个算法有零个或多个输入。大多数算法输入参数是必要的，但对于较简单的算法，如计算 1+2 的值，不需要任何输入参数，因此算法的输入可以是零个。

(5) 输出性：一个算法有一个或多个输出。算法用于某种数据处理，如果没有输出，这样的算法是没有意义的，这些输出是同输入有着某些特定关系的量。

**说明**：算法和程序是有区别的，程序是指使用某种计算机语言对一个算法的具体实现，即具体要怎么做，而算法侧重于对解决问题的方法描述，即要做什么。算法必须满足有限性，而程序不一定满足有限性，如 Windows 操作系统在用户没有退出、硬件不出现故障以及不断电的条件下理论上是可以无限时运行的，所以严格上讲算法和程序是两个不同的概念。当然算法也可以直接用计算机程序来描述，这样算法和程序就是一回事了，本书就是采用这种方式。

**【例 1.2】** 有下列两段描述。

描述 1：

```
void exam1()
{   int n;
    n = 2;
    while (n % 2 == 0)
        n = n + 2;
    printf(" % d\n",n);
}
```

描述 2：

```
void exam2()
{   int x,y;
    y = 0;
    x = 5/y;
    printf(" % d, % d\n",x,y);
}
```

这两段描述均不能满足算法的特征，试问它们违反了算法的哪些特征？

**解**：描述 1 是一个死循环，违反了算法的有限性特征。描述 2 出现除零错误，违反了算法的可行性特征。

## 1.1.2　算法描述

描述算法的方式很多，有的采用类 Pascal 语言，有的采用自然语言伪码。本书采用 C/C++语言来描述算法的实现过程，通常用 C/C++函数来描述算法。

以设计求 1+2+…+$n$ 值的算法为例说明 C/C++语言描述算法的一般形式，该算法如图 1.2 所示。

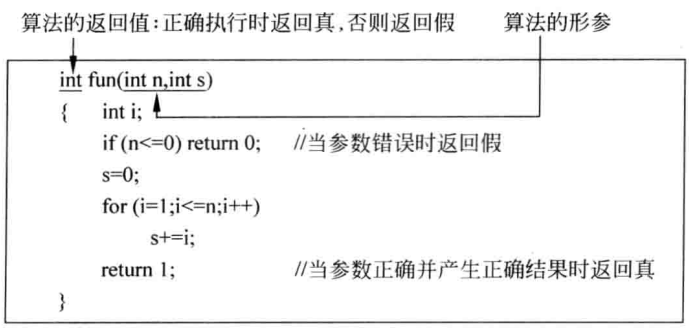



图 1.2　算法描述的一般形式

通常用函数的返回值表示算法能否正确执行，有时当算法只有一个返回值或者返回值可以区分算法是否正确执行时，用函数返回值来表示算法的执行结果，另外还可以带有形参表示算法的输入输出。任何算法(用函数描述)都是被调用的(在 C/C++语言中除 main 函

数外任何一个函数都会被其他函数调用，如果一个函数不被调用，这样的函数是没有意义的）。在C语言中调用函数时只有从实参到形参的单向值传递，执行函数时若改变了形参而对应的实参不会同步改变。例如，设计以下主函数调用上面的fun函数：

```
void main()
{   int a = 10,b = 0;
    if (fun(a,b)) printf(" % d\n",b);
    else printf("参数错误\n");
}
```

执行时发现输出结果为0，因为$b$对应的形参为$s$，fun执行后$s=55$，但$s$并没有回传给$b$。在C语言中可以用传指针方式来实现形参的回传，但增加了函数的复杂性。为此C++语言中增加了引用型参数的概念，引用型参数名前需“&”，表示这样的形参在执行后会将结果回传给对应的实参。上例采用C++语言描述算法如图1.3所示。

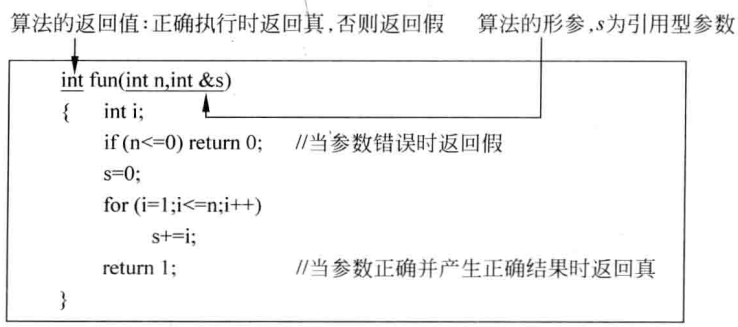



图1.3 带引用型参数的算法描述一般形式

当将形参$s$改为引用类型的参数后，执行时main函数的输出结果就正确了即输出55。由于C语言不支持引用类型，C++语言支持引用类型，所以本书算法描述语言为C/C++语言。需要注意的是，在C/C++语言中，数组本身就是一种引用类型，所以当数组作为形参需要回传数据时，其数组名之前不需要加“&”，它自动将形参数组的值回传给实参数组。

算法中引用型参数的作用如图1.4所示，在设计算法时，如果某个形参需要将执行结果回传给实参，则将该形参设计为引用型参数。带有引用型参数的程序不能在Turbo C中运行，可以在BC++、Visual C++、Dev C++等编译环境中运行。

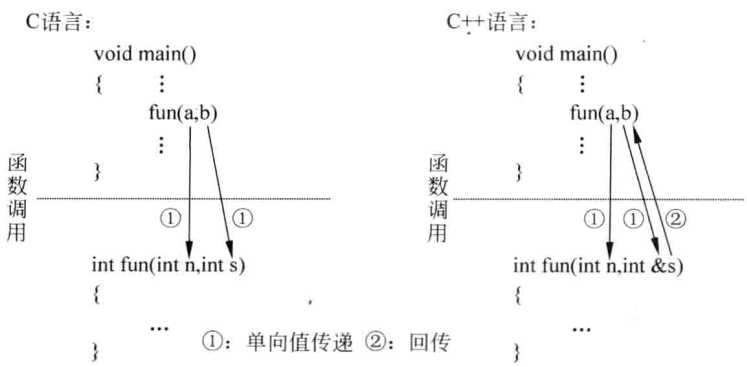



图1.4 算法中引用型参数的作用

**说明**：C语言没有提供实参到形参的双向传递，可以说是C语言的一个缺陷，在很多计算机语言中都改进了这一点，例如在Visual Basic语言中，函数形参需要指定是ByVal(单向值传递)还是ByRef(双向传递)。在C++中提供了引用类型，通常将函数形参定义为引用类型以实现实参到形参的双向传递。

### 1.1.3 算法和数据结构

算法与数据结构既有联系又有区别。

数据结构是算法设计的基础。算法的操作对象是数据结构，在设计算法时，通常要构建适合这种算法的数据结构。数据结构设计主要是选择数据的存储方式，如确定求解问题中的数据采用数组存储还是采用链表存储等。算法设计就是在选定的存储结构上设计一个满足要求的好算法。

另外，数据结构关注的是数据的逻辑结构、存储结构以及基本操作，而算法更多的是关注如何在数据结构的基础上解决实际问题。算法是编程思想，数据结构则是这些思想的逻辑基础。

### 1.1.4 算法设计的基本步骤

算法是求解问题的解决方案，这个解决方案本身并不是问题的答案，而是能获得答案的指令序列即算法，通过算法的执行获得求解问题的答案。算法设计是一个灵活的充满智慧的过程，其基本步骤如图1.5所示，各步骤之间存在循环反复的过程。

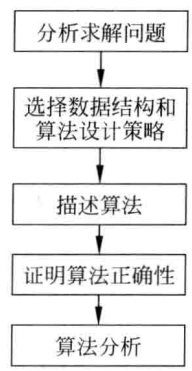


图1.5 算法设计的基本步骤

(1) 分析求解问题，确定求解问题的目标(功能)，给定的条件(输入)和生成的结果(输出)。

(2) 选择数据结构和算法设计策略，设计数据对象的存储结构，因为算法的效率取决于数据对象的存储表示。算法设计存在通用技术，如分治法、动态规划和回溯法等，需要针对求解问题选择合适的算法设计策略。

(3) 描述算法，在构思和设计了一个算法后，必须清楚准确地将所设计的求解步骤记录下来，即描述算法。

(4) 证明算法正确性，算法的正确性证明与数学证明有类似之处，因而可以采用数学证明方法。但用纯数学方法证明算法的正确性，不仅耗时，而且对大型软件开发也不适用。一般而言，为所有算法都给出完全的数学证明并不现实。因而选择那些已知是正确的算法，自然能大大减少出错的机会。本书介绍的大多数算法都是经典算法，其正确性已被证明，它们是实用和可靠的，书中主要介绍这些算法的设计思想和设计过程。

(5) 算法分析，同一问题的求解算法可能有多种，通过算法分析找到好的算法。一般来说，一个好的算法应该比同类算法的时间和空间效率高。

## 1.2 算法分析

计算机资源主要包括计算时间和内存空间。算法分析是分析算法占用计算机资源的情况。所以算法分析的两个主要方面是分析算法的时间复杂度和空间复杂度，其目的不是分

析算法是否正确或是否容易阅读，而主要是考查算法的时间和空间效率，以求改进算法或对不同的算法进行比较。

那么如何评价算法的效率呢？通常有两种衡量算法效率的方法：事后统计法和事前分析估算法。前者存在以下缺点：一是必须执行程序，二是存在其他因素掩盖算法本质的情况。所以下面均采用事前分析估算法来分析算法效率。

## 1.2.1 算法时间复杂度分析

### 1. 时间复杂度分析概述

一个算法用高级语言实现后，在计算机上运行时所消耗的时间与很多因素有关，如计算机的运行速度、编写程序采用的计算机语言、编译产生的机器语言代码质量和问题的规模等。在这些因素中，前 3 个都与具体的机器有关。撇开这些与计算机硬件、软件有关的因素，仅考虑算法本身的效率高低，可以认为一个特定算法的“运行工作量”的大小，只依赖于问题的规模（通常用整数量 $n$ 表示），或者说，它是问题规模的函数。

一个算法是由控制结构（顺序、分支和循环 3 种）和原操作（指固有数据类型的操作）构成的，算法的运行时间取决于两者的综合效果。例如，如图 1.6 所示是算法 Solve，其中形参 $a$ 是一个 $m$ 行 $n$ 列的数组，当是一个方阵（$m=n$）时求主对角线所有元素之和并返回 1，否则返回 0，从中看到该算法由 4 部分组成，包含两个顺序结构、一个分支结构和一个循环结构。

算法的执行时间主要与**问题规模**有关，例如数组的元素个数、矩阵的阶数等都可作为问题规模。算法执行时间是算法中所有语句的执行时间之和，显然与算法中所有语句的执行次数成正比。为了客观地反映一个算法的执行时间，可以用算法中基本语句的执行次数来度量，算法中的**基本语句**是执行次数与整个算法的执行次数成正比的语句，它对算法执行时间的贡献最大，是算法中最重要的操作。通常基本语句是算法中最深层循环内的语句，如图 1.6 的算法中 s+=a[i][i]就是该算法的基本语句。

```
void Solve(double a[][MAX],int m,int n,double &s)
{
    int i;
    s = 0;                          ——— 顺序结构

    if (m != n)
        return 0;                   ——— 分支结构

    for (i = 0; i < m; i++)
        s +=a[i][i];                ——— 循环结构

    return 1;                       ——— 顺序结构
}
```

图 1.6　一个算法的组成

设算法的问题规模为 $n$，以基本语句为基准统计出的算法执行时间是 $n$ 的函数，用 $f(n)$ 表示，对于如图 1.6 所示的算法，当 $m=n$ 时，算法中 for 循环内的语句为基本语句，它恰好执行 $n$ 次，所以有 $f(n)=n$。