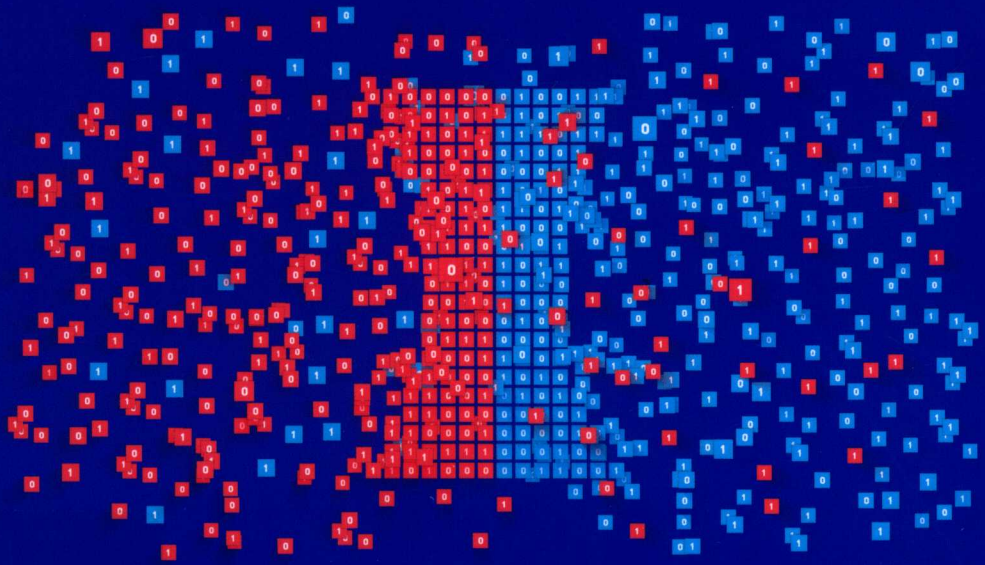
提供相关源代码下载

算法基础

打开程序设计之门

梁冰 冯林 刘胜蓝 / 编著



中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

算法基础

打开程序设计之门

梁冰 冯林 刘胜蓝 / 编著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

算法是一系列解决问题的清晰指令，是程序设计的灵魂。同一问题可采用不同的算法解决，而一个算法的优劣将直接影响程序的执行效率。

本书以 ACM 程序设计竞赛的题目为基础，详细介绍一些常用的算法以及相关的理论知识，主要内容包括高级数据结构、字符串、动态规划进阶算法、图论高级算法、经典算法问题、组合数学、计算几何、组合游戏论。

本书适合计算机专业的学生以及对程序设计竞赛感兴趣的读者阅读。

本书提供源代码下载，读者可登录华信教育资源网（www.hxedu.com.cn）免费注册后下载。

图书在版编目（CIP）数据

算法基础：打开程序设计之门 / 梁冰，冯林，刘胜蓝编著. —北京：电子工业出版社，2019.1
ISBN 978-7-121-35868-5

Ⅰ. ①算… Ⅱ. ①梁… ②冯… ③刘… Ⅲ. ①程序设计 Ⅳ. ①TP311.1

中国版本图书馆 CIP 数据核字（2018）第 296484 号

责任编辑：田宏峰　　特约编辑：李秦华
印　　刷：山东华立印务有限公司
装　　订：山东华立印务有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×980　1/16　印张：17.5　字数：392 千字
版　　次：2019 年 1 月第 1 版
印　　次：2019 年 1 月第 1 次印刷
定　　价：69.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：tianhf@phei.com.cn。

前　言

这是一本关于算法的教程。算法是一系列解决问题的清晰指令，可以说它是程序设计的灵魂。同一问题可用不同的算法解决，而一个算法的质量优劣将影响程序的执行效率。算法分析的目的在于选择合适算法和改进算法。评价一个算法的好坏主要是通过算法运行的时间长短和占用空间的大小来评估的。对于计算机专业或者爱好计算机的人士来说，无论学习还是工作，或多或少都会应用一些算法的知识。而目前国内外大型互联网公司在招聘时的笔试和面试都以算法为主。可见，算法的重要性是不言而喻的。

ACM/ICPC（ACM International Collegiate Programming Contest）是一项由美国计算机协会主办的，旨在展示大学生创新能力、团队精神和在压力下编写程序、分析和解决问题能力的年度竞赛。ACM 程序设计竞赛的题目强调算法的高效性与正确性。参赛选手只有编写出能够在规定时间内运行完成若干组数严格的测试数据，并且结果全部正确的程序才能得到分数。本书将以 ACM 程序设计竞赛的题目为基础，介绍一些经典的算法。

本书的目的是将更多对计算机算法感兴趣，但又苦于无从入手的读者带进程序设计的大门，让刚迈入大学校门的学生学会使用 C++语言解决简单的问题。本书主要介绍高级数据结构、字符串、动态规划、图论、组合数学方面的经典算法，相信当读者掌握了这些内容之后，会对算法和程序设计有一个新层次的认识，并会产生浓厚的兴趣。对于每个算法，本书都有图文并茂的讲解；在每章节的最后，都有针对该部分知识点的例题讲解，每道例题都是国内外著名程序在线判题系统中的原题，而且对于每道例题，都会从理解题意开始，详细讲解解题的思路，并附有完整的可以正确通过测试样例的代码，供读者研究学习。除了例题，在每章的最后还有一些练习题供读者巩固学到的知识，如果读者对这些习题仍感觉无从下手，可以参考每道练习题后附带的思路分析来帮助整理解题思路。

大连理工大学是在全国高校中较早倡导并开展创新创业教育的学校。自 1984 年以来，学校大力开展以突出创新创业实践为特色的创新创业教育。1995 年，在全国率先成立以学生创新创业教育为主体的教学改革示范区——创新教育实践中心，开展创新创业教育课程体系、教学内容、教学方法、教学模式等方面的改革，探索与之配套的管理运行机制，将创造性思维与创新方法融入教学实践中，在课堂教学中树立“CDIO 工程教育”新理念，倡导“做中学”，在实践环节构建了“个性化、双渠道、三结合、四层次、多模式”的创新教育实践教学新体系，取得了一系列成果，在全国高校产生了很大的影响。“创造性思维与创新方法”和“创新教育基础与实践（系列）”课程分别被评为国家级精品资源开放课程。“大学生程序设计竞赛初级教材”是“创新教育基础与实践”系列课程的核心课程，是面向大连理工大学 ACM 创新实践班的学生开设的。

此外，本书在撰写过程中，除了参考文献和正文中标出的引用来源，还参考了国内外的相关研究成果和网站资源，但没有一一列出，在此感谢所涉及的所有单位、专家和研究人员。

因编者水平有限，书中的错误和不足之处在所难免，欢迎广大读者来信批评指正，提出宝贵意见，帮助我们不断地完善本书。

编　者

2018 年 12 月

目　录

第 1 章

高级数据结构

本章介绍一些高级数据结构。正确地选取数据结构能大大提高程序的效率。基础数据结构，如线性表（栈、队列、链表等）、二叉树、图等，是高级数据结构的基础。

堆是一种常见的数据结构。堆排序是利用堆这种数据结构所设计的一种选择排序。堆可以实现优先队列。树状数组可以简单高效地求得区间和。平衡二叉树是一棵平衡的二叉树，能让二叉树的操作维持在 $O(\log_2 N)$左右。Treap 通过随机数来优化二叉查找树（Binary Search Tree）防止其退化。Splay 树通过其特有的 Splay 操作来维持平衡。左倾堆是一种可并堆，具有神奇的“左倾”性质。

1.1 堆

本节介绍一种常见的数据结构——二叉堆（简称堆），通过图文讲解和代码实现来了解这个重要的数据结构。

1.1.1 堆的定义

堆（Heap）是一棵完全二叉树。其最重要的性质就是“儿子”的值一定不小于（或大于）“父亲”的值（分别称为小顶堆和大顶堆，下文使用大顶堆）。应用场景包括堆排序和优先队列等。用数组存堆时，对于任意一个节点 x，其左、右子节点的标号分别为 $2x+1$ 和 $2x+2$。

堆的逻辑结构和存储结构如图 1.1 所示。

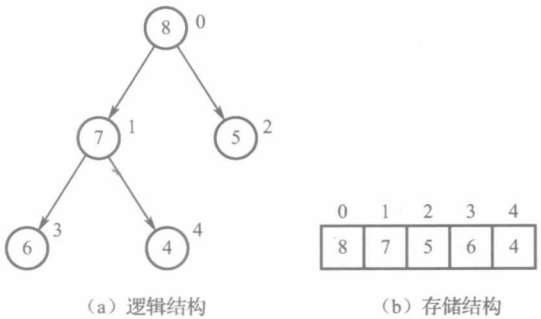


图 1.1 堆的逻辑结构和存储结构

1.1.2 建堆

建堆的核心内容是调整堆，使二叉树满足堆的定义（每个节点的值都不大于其父节点的

值）。调堆的过程应该从最后一个非叶子节点开始，一直调整到根节点。

堆排序过程示例如图 1.2 所示。

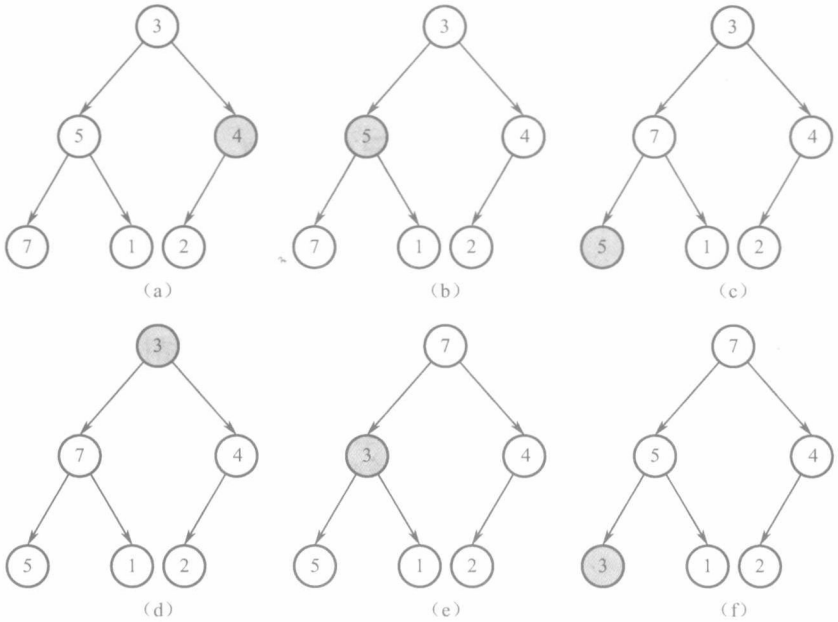


图 1.2　堆排序过程示例

1.1.3　堆排序算法

设堆有 n 个元素，每一次调整堆，堆顶位置将得到堆的最大值，然后将堆顶元素和最后一个元素互换，对前 $n-1$ 个元素继续调整，直到整个堆有序为止。堆排序时间复杂度为 $O(\log_2 N)$。堆排序是不稳定排序。

代码实现如下所述。

```
#include <cstdio>

void heap_adjust(int arr[], int father, int n)
{
    //调整为大顶堆
    int child = father * 2 + 1;                    // 左子节点
    int tmp = arr[father];
    while (child < n) {
        //如果该节点有两个子节点，则取其中较大的一个，否则取左子节点
        if (child + 1 < n && arr[child] < arr[child + 1]) child++;
        if (arr[father] >= arr[child]) break;
```

```
12              //如果较大的子节点小于该节点，则不需要继续调整，退出
13              arr[father] = arr[child];              //否则交换该节点和较大的子节点，继续向下调整
14              father = child;
15              child = father * 2 + 1;
16              arr[father] = tmp;
17          }
18      }
19
20      void build_heap(int arr[], int n)
21      {
22          //建堆时从非叶子节点开始调整
23          for (int i = (n - 1) / 2; i >= 0; --i) {
24              heap_adjust(arr, i, n);
25          }
26      }
27
28      void heap_sort(int arr[], int beg, int end)
29      {
30          //堆排序，先建堆
31          //然后每一次交换未调整好的最后一个元素和第一个元素
32          build_heap(arr + beg, end - beg);
33          for (int tmp, i = end - 1; i > beg; --i) {
34              tmp = arr[i]; arr[i] = arr[0]; arr[0] = tmp;
35              heap_adjust(arr + beg, 0, i);
36          }
37      }
38
39      int main()
40      {
41          int arr[100];
42          int n; scanf("%d", &n);
43          for (int i = 0; i < n; ++i) scanf("%d", &arr[i]);
44          heap_sort(arr, 0, n);
45          for (int i = 0; i < n; ++i) printf("%d ", arr[i]);
46          return 0;
47      }
```

1.2　树状数组

树状数组（Binary Indexed Tree，BIT）能够高效地求序列区间和。树状数组的实现简单，

巧妙地运用了二进制的思想。

1.2.1 树状数组的定义

给定一个数组进行两个操作：一是更新某点的值；二是求某段区间的和。对于普通的数组，单点更新的复杂度为 $O(1)$，求区间和的复杂度为 $O(n)$，而树状数组能够把单点更新和区间求和的复杂度都变为 $O(n\log n)$。

设数组 $A[]$为原数组，定义 $C[i]=A[i-2^k+1]+\cdots+A[i]$。其中，$k$ 为 i 用二进制表示时的末尾 0 的个数，如 $C[1]=A[1]$，$C[2]=A[1]+A[2]$，$C[3]=A[3]$，$C[4]=A[1]+A[2]+A[3]+A[4]$，…。也就是说，$C[i]$就是从 $A[i]$开始前 2^k 项的和，如图 1.3 所示。

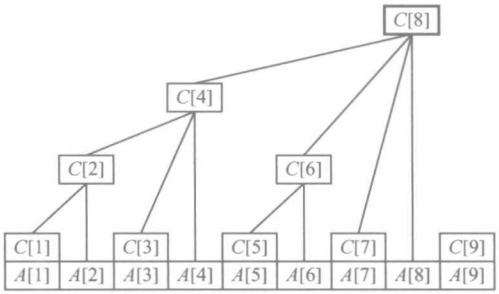


图 1.3 树状数组

1.2.2 树状数组的实现和使用

k 为 x 用二进制表示时末尾 0 的个数，对于 2^k，有一种快速的求解方法：

```
1    int lowbit(int x) {
2        return x & (-x);
3    }
```

当修改某一点的值时，只需要修改某一点的所有父节点就可以了。对于一个节点 i 来说，它的父节点的编号为 i+lowbit(i)。因此对于单点修改的函数为

```
1    //在 position 处加 value，len 是数组长度
2    void change(int c[], int position, int value, int len) {
3        while (position <= len) {
4            c[position] += value;
5            position += lowbit(position);
6        }
7    }
```

前 n 个数的和记为 sum(n)，已知 $C[i]$是从 i 开始向前 lowbit(i)个数的和，所以

sum(n)=C[n]+sum[n−lowbit(n)]。

```
1    //求前 n 个数的和
2    int sum(int c[], int n) {
3        int answer = 0;
4        while (n > 0) {
5            answer += c[n];
6            n -= lowbit(n);
7        }
8        return answer;
9    }
```

区间[start,end]的区间和可以通过 sum(end)−sum(start)来求得。树状数组也可以进行区间增减更新和单点查询操作。A[]是原数组，构造差分数组 D[]，令 $D[1]=A[1]$，$D[i]=A[i]-A[i-1]$($i>1$)，则数组 A 是数组 D 的前缀和，即 $A[i]=D[1]+D[2]+\cdots+D[i]$。这样求 A 的单点值就是求 D 的区间和。更新 A 某段区间[start,end]的值只需要更改 D[start]和 D[end+1]的值就可以了。这两个操作可以通过构造 D 的树状数组求得。

1.2.3　例题讲解

例 1-1　Sort it

Time Limit: 2000/1000 ms (Java/Others)　　Memory Limit: 32768/32768 KB (Java/Others)

题目描述：

给定一个由 n 个不同的整数组成的序列，通过交换相邻的两个数，使得序列变成上升的。问最少需要交换多少次，如“1 2 3 5 4”，需要进行一次操作，交换 5 和 4。

输入：

输入包含多组数据，每一组数据包含两行，第一行为一个正整数 n（$n\leqslant 1000$），第二行为从 1 到 n 的 n 个整数的一个序列。

输出：

输出为 1 行，输出最少需要交换的次数。

样例输入：

3
1 2 3
4
4 3 2 1

样例输出：

0
6

题目来源：HDOJ 2689。

解题思路：

题目可以转化成求数组中逆序对的数量。

逆序对：设数组 A 为一个有 n 个数字的有序集（n>1），其中的数字均不相同，如果存在正整数 i，j，使得 $1 \leq i < j \leq n$ 而且 $A[i] > A[j]$，则<$A[i], A[j]$>这个有序对称为 A 的一个逆序对。

对于每一个数字，求它前面有多少个数字大于它，即该数字的贡献，所有数字的贡献和即逆序对的数量。对于 $A[i]=x$，在树状数组的 x 处加 1，然后求 sum(x)就是在 x 前面有多少个数小于等于 x，那么 i–sum(x)就是 $A[i]$的贡献。从左到右对于每一个数边插入边求和。

题目实现：

```
#include <cstdio>
#include <cstring>
using namespace std;

const int N = 1000;

int lowbit(int x) {
    return x & (-x);
}

void change(int c[], int position, int value, int len) {
    //…
}

int sum(int c[], int n) {
    //…
}

int main()
{
    int n;
    int c[N];                                   //树状数组
    while (~scanf("%d", &n)) {                  //序列包含多少个数字
        memset(c, 0, sizeof(c));
        int x;
        int answer = 0;
        for (int i = 1; i <= n; ++i) {
            scanf("%d", &x);                    //读入每一个数字
            change(c, x, 1, n);                 //在树状数组 x 处加 1
            answer += i - sum(c, x);            //把每一个数字的贡献相加
```