HZ BOOKS
华章教育

本书用于
ACM/ICPC、Google Code Jam、TopCoder
百度之星程序设计大赛
等程序设计竞赛的训练

程序设计解题策略

大学程序设计课程与竞赛训练教材

Programming Strategies Solving Problems

for Collegiate Programming Contest and Education

吴永辉 王建德 编著

机械工业出版社
China Machine Press

程序设计解题策略

大学程序设计课程与竞赛训练教材

Programming Strategies for Solving Problems

for Collegiate Programming Contest and Education

吴永辉 王建德 编著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

程序设计解题策略 / 吴永辉，王建德编著 . —北京：机械工业出版社，2015.1

ISBN 978-7-111-48831-6

I. 程… II. ①吴… ②王… III. 程序设计 IV. TP311.1

中国版本图书馆 CIP 数据核字（2014）第 293430 号

本书在数据结构和算法设计的基础上，从树型数据关系、图型数据关系、数据关系的构造策略、数据统计的二分策略、动态规划的优化策略、计算几何的应对策略及博弈问题的应对策略七个方面，具体介绍了 49 种解题策略和重要算法。全书结合国内外多年程序设计竞赛的经典例题，精选出 100 道实验范例，每道实验范例均注明了试题来源和在线测试网址，帮助读者更加深入地了解和掌握编程解题策略。另外，所有试题的原版描述和大部分试题的测试数据可登录华章网站下载。

本书既可作为程序设计竞赛的培训教材，也可作为高等院校计算机及相关专业程序设计的相关教材。

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）

责任编辑：李　艺　　责任校对：董纪丽

印　　刷：北京瑞德印刷有限公司　　版　　次：2015 年 2 月第 1 版第 1 次印刷

开　　本：185mm×260mm　1/16　　印　　张：32.25

书　　号：ISBN 978-7-111-48831-6　　定　　价：79.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88378991　88361066　　投稿热线：（010）88379604

购书热线：（010）68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

前　言

策略即计策和谋略，指一种总体的行为方针和行事方法，即一种可以实现目标的方案集合，而非纠缠于细枝末节的雕虫小技。程序设计的解题策略指的是编程解题过程中所采取的一种基本方法，是对解题方法的综合性、智能性和个性化的认识。尤其是当面对非标准、非模式化的问题时，就更需要发挥创造性思维，求索应对策略和解题艺术。正如古人所言“术谋之人以思谟为度，故能成策署之奇，而不识遵法之良”。

对于程序设计竞赛选手的培养，教师应注重在两个方面系统地训练选手：①程序设计竞赛的知识体系；②程序设计的解题策略。

对于程序设计竞赛的知识体系，可以用“算法 + 数据结构 = 程序”这一著名公式来概括。为此，这两年我们先后在机械工业出版社出版了两本专著（《数据结构编程实验》和《算法设计编程实验》），其中《数据结构编程实验》还由中国台湾碁峰出版社以《提升程式設計的資料結構力：國際程式設計競賽之資料結構原理、題型、解題技巧與重點解析》为书名出版。

上述两本书和本书一起，构成了对程序设计竞赛选手进行“程序设计竞赛的知识体系”和“程序设计的解题策略”系统训练的一个完整系列，本系列也可以作为大学的程序设计实验课程的系列教材。这三本书包含的知识结构和实验选题没有重复且互相联系：前两本书在系统讲述程序设计竞赛知识体系结构的同时，也涉及一些初浅的解题策略，但主要是从知识的角度谈解题方法；而本书所讲述的解题策略虽是在深入数据结构和算法设计知识的基础上展开，但侧重从行为特征的角度谈解题方法。所以，这三本书的知识层次和论述角度有区别、有联系、有递进关系。

为了使读者对编程解题的策略有一个全面了解，我们将这些年国际、国内程序设计竞赛的经典例题进行分门别类，从七个大的方面讨论解题策略。

（1）利用树型数据关系解题的七种基本策略

利用划分树求解整数区间内第 k 大的值；利用最小生成树原理和拓展形式（最小 k 度限制生成树和次小生成树）求解带权无向连通图中最佳边权和的生成树；利用一维线段树计算和维护区间的最值（最大或最小值）和总量；利用两种改进型的二叉查找树——伸展树和红黑树来优化动态集合的操作；利用动态树计算和维护一个带权森林；利用左偏树实现优先队列的合并；利用跳跃表实现树的功能。

（2）利用图型数据关系解题的五种基本策略

将图论问题转化成网络流的计算；将一般图转换成二分图，利用匹配算法提高计算效率；通过分层图思想将干扰因素细化为若干状态，通过层的连接将状态联系起来，最终找到算法；从信息原型中挖掘出平面图特征，洞悉有向边蕴含的偏序关系，清晰思路、简化算法；从挖掘和利用图论模型性质的三个思考角度出发，提出选择图论模型和优化算法的三种基本策略（“定义法”、“分析法”、“综合法”）。

（3）数据关系上的构造的三种基本策略

选择逻辑结构的两个基本原则（利用“可直接使用”的信息和不记录“无用”的信息）；

（续）

序号	解题策略和重要算法	所在章节
30	DP 中减少状态总数的基本策略	第 5 章　动态规划上的优化策略
31	DP 中减少每个状态决策数的基本策略	
32	DP 中减少状态转移时间的基本策略	
33	基于状态压缩的插头 DP	
34	模拟退火算法	第 6 章　计算几何上的应对策略
35	三分法	
36	圆重合其他几何图形时的剖分策略	
37	三角剖分	
38	梯形剖分	
39	矩形切割思想	
40	利用极大化思想解决最大子矩形问题	
41	合理组合基本几何运算	
42	利用动态博弈思想判断输赢	第 7 章　博弈类问题的应对策略
43	巴什博弈	
44	威佐夫博弈	
45	尼姆博弈	
46	k 倍动态减法游戏	
47	尼姆博弈的四种扩展形式	
48	SG 函数	
49	使用 surreal number 应对组合游戏	

这些策略和算法有的源于程序设计领域的经典成果，但更多的是源于复旦大学程序设计竞赛队的成员在长期编程实践中积淀的策略经验。我们对此进行了认真的整理和提炼，使之由默会知识变成明言知识。书中所述的每个解题策略和算法原理都有必要的分析和证明，定理证明大多采用初等数学的分析方法，公式推导尽可能做到浅显和详细，并给出了计算时间复杂度的详细分析。为了帮助读者理解，对其中一些复杂的解题策略和算法附加了图示，使其过程更加具体、直观和形象。

“程序设计解题策略”是一门“实践出真知”的课程，需要大量引入案例教学，通过模拟或者重现现实生活中的一些场景，让读者置身于问题情境之中，通过思考、讨论和上机编程来学习解题策略。为此，我们在浩如烟海的各类程序设计竞赛试题中精选出 100 道试题作为实验范例，启发性地引出相应的解题策略，为读者学习解题策略的相关知识创设丰富有趣的问题背景。每个实验范例不仅有知识要点阐述和详尽的试题解析，而且还给出了写有详细注释的参考程序；每个常用的解题策略都以经典模型为支撑，每个定理概念都有详尽的说明和推导，并使用大量图表增加直观性和可读性。本书尽力将经典模型推广到具有相同性质的一类问题，让读者熟悉一个经典模型就相当于掌握了解决一类问题的方法。有时还有意在经典模型外加入其他因素，引导读者对被套用的经典模型进行适当修改，加入独特的属性，或者运用已知模型或方法来分析信息原型的属性，在此基础上创造出具有新意的模型或方法。为了让读者能够检验自编程序的正确性和效率，每道题都注明了试题来源和在线测试网址。另外，可登录华章网站（www. hzbook. com）下载所有试题的原版描述和大部分试题的测试数据等资料。

本书试题的在线测试地址主要有：

在线评测系统	简称	网址
北京大学在线评测系统	POJ	http://poj.org/
浙江大学在线评测系统	ZOJ	http://acm.zju.edu.cn/onlinejudge/
UVA 在线评测系统	UVA	http://uva.onlinejudge.org/ http://livearchive.onlinejudge.org/
Ural 在线评测系统	Ural	http://acm.timus.ru/
SGU 在线评测系统	SGU	http://acm.sgu.ru/
杭州电子科技大学在线评测系统	HDOJ	http://acm.hdu.edu.cn/

编程解题离不开建模和解题方法，解题的成功取决于选择正确的数学模型和解题方法，而数学模型和解题方法的正确性源于采用适宜的解题策略。建议读者在学习各种解题策略时，注意以下三个方面的问题。

1）培养良好的认知结构。注重相关理论的学习，并通过不断应用来强化理论知识，形成一个脉络分明、纵横交错的知识网络。同时注意把一些常用的解题技巧和变换方法放在记忆库里，把同类问题“存储在一起”，使知识条理化。

2）提高解题能力。平时要多看书和多解题，从书中和编程实践中寻找再发现、再创造的契机。既要注意运算结果的正确性，也要注意知识产生的过程性（概念、法则被概括的过程，数据关系被抽象的过程，解题策略被探索的过程）。要细化书本知识，如同电视屏幕中体育大赛慢镜头式的分解，真正使自己学有所得。

3）注重解题策略的归类分析。就某个方面的解题策略而言，其形成过程是可以逻辑化、模式化的。因此，要多考虑将解题策略归类，可以选取一些具有代表性的例题，进行系统的、集中的分类解题策略训练，形成一套局部范围内的逻辑化、模式化的解题策略方案。

需要说明的是，张一博、舒天民、王禹、金阳华等同学精心编写了所有程序，每道程序都通过了严格的测试验证，多次修改、精益求精，特别是张一博和王禹同学还对本书的结构和内容提出了系统的、建设性的建议。这几位同学为本书的出版付出了辛勤劳动，做出了不可或缺的贡献，在此向他们表示由衷的感谢。另外，衷心感谢杭州电子科技大学刘春英老师为本书的部分试题在 HDOJ 上提供了在线测试平台；衷心感谢中国台湾的老师和同学在使用本书书稿过程中提出的宝贵意见和建议。

由于时间和水平所限，书中肯定夹杂了不少缺点和错误，表述不当和笔误更是在所难免，热切欢迎学术界同仁或读者赐正。如果你在阅读中发现了问题，请通过书信或电子邮件告诉我们，以便及时整理成勘误表放在本书的网站上，供广大读者查询更正。我们更期望读者对本书提出建设性意见，以便再版时改进。我们的联系方式如下。

通信地址：上海市邯郸路 220 号复旦大学计算机科学技术学院　吴永辉

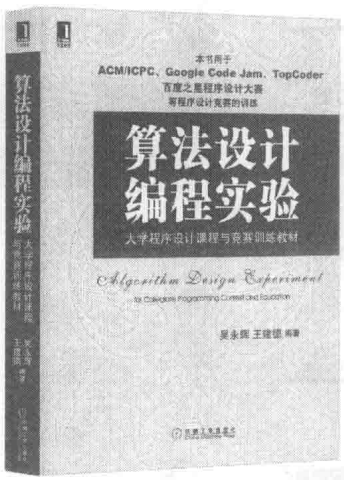
邮编：200433

Email：yhwu@fudan.edu.cn

吴永辉　王建德

2014 年 10 月

推荐阅读

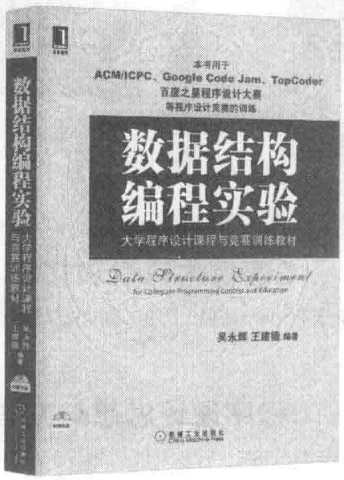


算法设计编程实验：大学程序设计课程与竞赛训练教材

作者：吴永辉 等 ISBN：978-7-111-42383-6 定价：69.00元

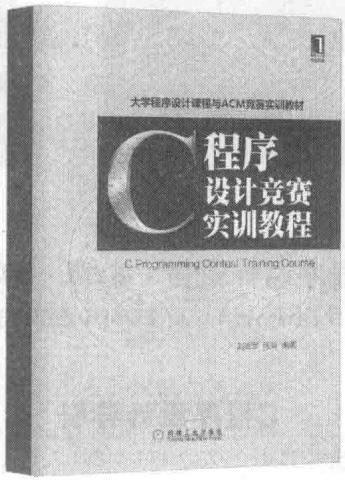
ACM-ICPC世界总决赛试题解析(2004~2011年)

作者：吴永辉 等 ISBN：978-7-111-39094-7 定价：55.00元



数据结构编程实验：大学程序设计课程与竞赛训练教材

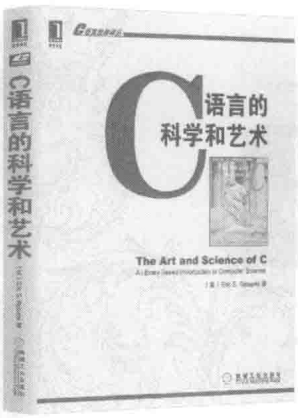
作者：吴永辉 等 ISBN：978-7-111-37395-7 定价：59.00元

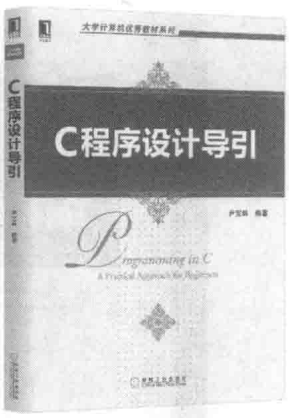


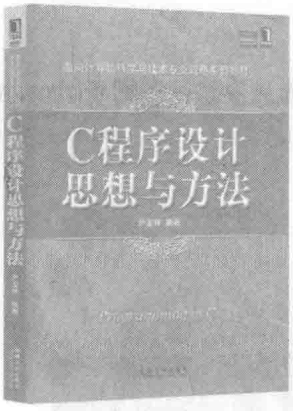
C程序设计竞赛实训教程

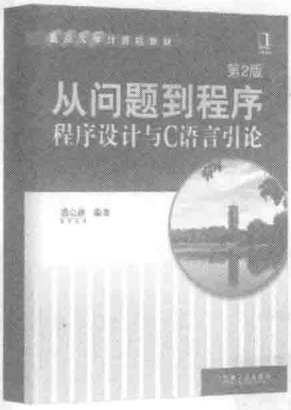
作者：刘高军 等 ISBN：978-7-111-38917-0 定价：29.00元

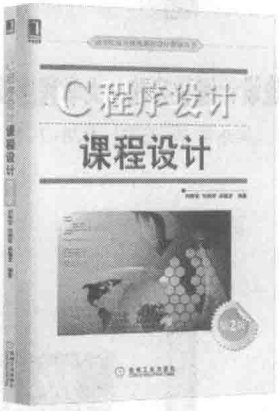
推荐阅读











C程序设计语言（第2版·新版）

作者：Brian W. Kernighan 等 ISBN：978-7-111-12806-0 定价：30.00元

C语言的科学和艺术

作者：Eric S. Roberts ISBN：978-7-111-34775-0 定价：79.00元

C程序设计导引

作者：尹宝林 ISBN：978-7-111-41891-7 定价：35.00元

C程序设计思想与方法

作者：尹宝林 ISBN：978-7-111-25495-9 定价：36.00元

从问题到程序——程序设计与C语言引论 第2版

作者：裘宗燕 ISBN：978-7-111-33715-7 定价：39.00元

C程序设计课程设计 第2版

作者：刘振安 等 ISBN：978-7-111-28541-0 定价：24.00元

目　录

第1章

利用树型数据关系的解题策略

树是一个具有层次结构的集合，一种限制前件数且没有回路的连通图。在现实生活和程序设计的竞赛试题中，许多问题的数据关系呈树型结构，因此有关树的概念、原理、操作方法和一些由树的数据结构支持的算法，一直受到编程者的重视，被广泛应用于解题过程。在本章里，我们将介绍利用树型数据关系解题的七种策略：

1）利用划分树求解整数区间内第 k 大值。

2）利用最小生成树及其扩展形式(最优比率生成树、最小 k 度生成树、次小生成树)计算有权连通无向图中边权和满足限制条件的最优生成树。

3）利用线段树计算区间。

4）利用改进型的二叉查找树(伸展树和红黑树)优化动态集合的操作。

5）利用动态树维护森林的连通性。

6）利用左偏树实现优先队列的合并。

7）利用跳跃表取代树结构，这种策略不失时空效率，容易编程实现。

前六种树型结构鲜见于一般数据结构教材，利用这些特殊类型的树可优化存储结构和算法效率。

1.1 利用划分树求解整数区间内第 k 大的值

如何快速求出(在 $\log_2 n$ 的时间复杂度内)整数区间[x, y]中第 k 大的值($x \leqslant k \leqslant y$)？注意，所谓区间指的是整数的序号范围，[$x$, y]指区间内第 x 个整数至第 y 个整数间的 $y-x+1$ 个整数，而不是指整数值本身。

人们通常会想到采用快速查找法查找整数区间的第 k 大值，但这种方法会破坏原序列且需要 $O(n)$的时间复杂度，即便使用平衡二叉树进行维护，虽每次查找时间复杂度降为 $O(\log_2 n)$，但由于丢失了原序列的顺序信息，也很难查找出某区间内的第 k 大值。

划分树主要是针对上述问题而设计的。划分树是一种基于线段树的数据结构，其基本思想就是将待查找区间划分成两个子区间：不大于数列中间值的元素被分配到左儿子的子区间，简称左子区间；大于数列中间值的元素被分配到右儿子的子区间，简称右子区间。

显然，左子区间的数小于右子区间的数。建树的时候要注意，对于被分到同一子树的元素，元素间的相对位置不能改变。例如构建整数序列[1 5 2 3 6 4 7 3 0 0]的划分树：

整数序列经排序后得到[0 0 1 2 3 4 5 6 7]，中间值为3。划分出下一层的左子区间[1 2 3 0 0]，中间值为1；下一层的由子区间[5 6 4 7 3]，中间值为5；以中间值为界，划分出当前层每个区间的下层左右子区间……依此类推，直至划分出的所有子区间含单个整数为止(见图 1.1-1)。

由图 1.1-1 可见，划分树是一棵完全二叉树，树叶为单元素区间。若数列含 n 个整数，则对应的划分树有$\lceil \log_2 n \rceil + 1$ 层。

查找的时候通过记录进入左子树的数的个数，确定下一个查找区间，直至查找范围缩小到单元素区间为止。此时，区间内的第 k 大值就找到了。

整个算法分两步：

1）建树——离线构建整个查询区间的划分树。

2）查找——在划分树中查找某个子区间中第 k 大值。

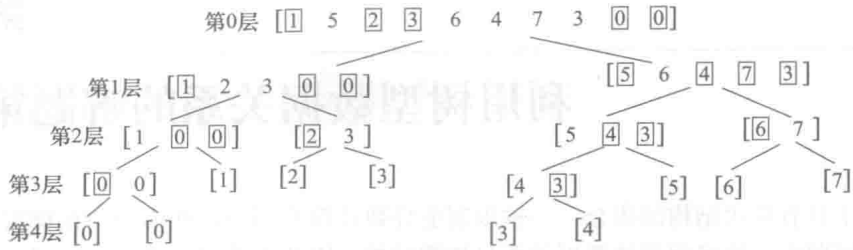


图 1.1-1　线框内的数被分配到下一层的左子区间

1.1.1　离线构建整个查询区间的划分树

在查询之前，我们先离线构建整个查询区间的划分树。建树过程比较简单，对于区间$[l, r]$，首先通过对原数列排序找到这个区间的中间值的位置 $\text{mid}\left(\text{mid}=\left\lfloor\frac{l+r}{2}\right\rfloor\right)$，不大于中间值的数划入左子树$[l, \text{mid}]$，大于中间值的数划入右子树$[\text{mid}+1, r]$。同时，对于第 i 个数，记录在$[l, i]$区间内有多少整数被划入左子树。然后继续对它的左子树区间$[l, \text{mid}]$和右子树区间$[\text{mid}+1, r]$递归建树，直至划分出最后一层的叶节点为止，显然，建树过程是自上而下的。

具体实现办法：

先将读入的数据排序成 sorted[]，取区间$[l, r]$的中间值 $\text{sorted}[\text{mid}]\left(\text{mid}=\left\lfloor\frac{l+r}{2}\right\rfloor\right)$，然后扫描$[l, r]$区间，依次将每个整数划分到左子区间$[l, \text{mid}]$或右子区间$[\text{mid}+1, r]$中去。注意，被分到每个子树的整数是相对有序的，即对于被分到每一棵子树里面的数，不改变它们以前的相对顺序。另外，在进行这个过程的时候，记录一个类似前缀和的东西，即 l 到 i 这个区间内有多少整数被划分到左子树。设：

tree$[p][i]$——第 p 层中第 i 位置的整数值，初始序列为 tree[0][]。

sorted[]——排序序列，即存储 tree[0][]排序后的结果。

toleft[][]——toleft$[p][i]$表示第 p 层前 i 个数里中有多少个整数分入下一层的左子区间。

lpos 和 rpos——下一层左子区间和右子区间的指针。

same——当前区间等于中间值的整数个数。

下面给出实现建树的具体代码。

```
void build(int l,int r,int dep)                    // 从 dep 层的区间[l,r]出发,自上而下构建划分树
{
    if(l==r)return;                                // 若划分至叶子,则回溯
    int mid=(l+r)>>1;                              // 计算区间的中间指针
    int same=mid-l+1;                              // 计算[l,r]的中间值被分入下层后左子区间的
                                                   // 个数 same
    for(int i=l;i<=r;i++) if(tree[dep][i]<sorted[mid])same--;
    int lpos=l;                                    // 下一层左子区间和右子区间的指针初始化
    int rpos=mid+1;
    for(int i=l;i<=r;i++)                          // 搜索区间[l,r]中的每个数
    {
      if(tree[dep][i]<sorted[mid])                 // 若 dep 层的第 i 个数据比中间值小,则被划入
// 下一层的左子区间;若相同于中间值,被划入下一层的左子区间,中间值被分入下层后左子区间的个数-1;
// 若大于中间值,则被划入下一层的右子区间
       tree[dep+1][lpos++]=tree[dep][i];
```

```
        else if(tree[dep][i] == sorted[mid]&&same > 0)
                { tree[dep+1][lpos++] = tree[dep][i];same--; }
            else tree[dep+1][rpos++] = tree[dep][i];
        toleft[dep][i] = toleft[dep][l-1] + lpos - l;     // 计算 dep 层的第 l 个数到第 i 个数放入下一层
                                                          // 左子区间的个数
    }
    build(l,mid,dep+1);                                   // 递归计算下一层的左子区间
    build(mid+1,r,dep+1);                                 // 递归计算下一层的右子区间
}
```

1.1.2　在划分树上查询子区间[l, r]中第 k 大的数

如何在大区间[L, R]里查询子区间[l, r]中第 k 大的数呢($L \leqslant l$, $r \leqslant R$)？我们从划分树的根出发，自上而下查找：

若查询至叶子($l == r$)，则该整数(tree[dep][l])为子区间[l, r]中第 k 大的数；否则区间[L, $l-1$]内有 toleft[dep][$l-1$]个整数进入下一层的左子树，区间[L, r]内有 toleft[dep][r]个整数进入下层的左子树。显然，在查询子区间[l, r]中有 cnt = toleft[dep][r] - toleft[dep][$l-1$]个整数进入了下一层的左子树。如果 cnt $\geqslant k$，则递归查询左子树(对应区间[L, mid])；否则递归查询右子树(对应区间[mid+1, R])。

难点是，如何在对应子树的大区间中计算被查询的子区间[newl, newr]？

若递归查询左子树，则当前子区间的左边是上一层区间[L, $l-1$]里的 toleft[dep][$l-1$] - toleft[dep][$L-1$]个整数，由此得到 newl = L + toleft[dep][$l-1$] - toleft[dep][$L-1$]；加上上一层查询区间[l, r]中的 cnt 个整数，newr = newl + cnt - 1(这里 -1 是为了处理边界问题，值得大家认真思索)，即在大区间[L, mid]里查询子区间[newl, newr]中第 k 大的值。

同理，若递归查询右子树，则[l, r]中有 $r-l-$ cnt 个整数进入右子树，需要计算其中第 $k-$ cnt 大的整数值。关键是如何计算被查询子区间的右指针 newr：

当上一层的[l, r]和[$r+1$, R]中有若干个整数被分配至下一层左子区间，其中[$r+1$, R]中有 toleft[dep][R] - toleft[dep][r]个整数位于左子区间右方(保持相对顺序不变)，因此查询子区间的右指针移至 r 右方的第 toleft[dep][R] - toleft[dep][r]个位置，即 newr = r + toleft[dep][R] - toleft[dep][r]。显然，左指针 newl = newr - ($r-l-$ cnt)。

下面给出实现查询的具体代码。

```
int query(int L,int R,int l,int r,int dep,int k)      // 从划分树的 dep 层出发,自上而下在大区间
                                                      // [L,R]里查询子区间[l,r]中第 k 大的数
{
  if(l == r)return tree[dep][l];                      // 若查询至叶子,则该整数为子区间[l,r]中
                                                      // 第 k 大的数
  int mid = (L+R) >>1;                                // 取大区间的中间指针
  int cnt = toleft[dep][r] - toleft[dep][l-1];        // 计算[l,r]中被划入下一层左子区间的整数个数
  if(cnt >= k)                                        // 递归查询左子树,对应的大区间为[L,mid],
                                                      // 小区间为[newl,newr],计算其中第 k 大的值
       int newl = L + toleft[dep][l-1] - toleft[dep][L-1];
       int newr = newl + cnt -1;
       return query(L,mid,newl,newr,dep+1,k);
  }
  Else                                                // 递归查询右子树,对应的大区间为[mid+1,R],
                                                      // 小区间为[newl,newr],计算其中第 k-cnt 大的值
  {
       int newr = r + toleft[dep][R] - toleft[dep][r];
       int newl = newr - (r - l - cnt);
```

```
            return query(mid+1,R,newl,newr,dep+1,k-cnt);
        }
    }
```

1.1.3　应用划分树解题

虽然划分树是一种基于线段树的数据结构，但被分到同一子树的元素间的相对位置不改变，因此将其用于查询子区间[l, r]中第 k 大的值是十分适合的。

【1.1.1　K-th Number】

【问题描述】

您为 Macrohard 公司的数据结构部门工作，现在被要求编写一个新的数据结构，该数据结构能在一个数组段中快速地返回第 k 个次序统计数据。

也就是说，给出一个数组 $a[1..n]$，包含了不同的整数，程序要回答一系列这样形式的问题 $Q(i, j, k)$："在数组 $a[i..j]$段中第 k 个数是什么，如果这一段是排好序的?"

例如，数组 $a=(1, 5, 2, 6, 3, 7, 4)$，问题是 $Q(2, 5, 3)$，数组段 $a[2..5]$是(5, 2, 6, 3)。如果对这一段进行排序，得(2, 3, 5, 6)，第3个数是5，所以这个问题的答案是5。

输入：

输入的第一行给出数组的大小 n 和要回答的问题数 $m(1\leqslant n\leqslant 100\,000,\ 1\leqslant m\leqslant 5000)$。

第二行给出回答问题的数组，即 n 个不同绝对值不超过 10^9 的整数。

后面的 m 行给出问题描述，每个问题由3个整数组成：i、j 和 $k(1\leqslant i\leqslant j\leqslant n,\ 1\leqslant k\leqslant j-i+1)$，表示相应的问题 $Q(i, j, k)$。

输出：

对每个问题输出答案——在已排好序的 $a[i..j]$段输出第 k 个数。

样例输入	样例输出
7 3 1 5 2 6 3 7 4 2 5 3 4 4 1 1 7 3	5 6 3

提示：本题海量输入，采用 C 风格输入－输出(scanf, printf)，否则超时。

试题来源：ACM Northeastern Europe 2004, Northern Subregion

在线测试地址：POJ 2104

试题解析

简述题意：给出一个长度为 n 的整数序列，要求你反复采用快速查找方法查找某个子序列中第 k 大的值。

显然，采用划分树查询是最高效的。方法是：

1）先离线构建整数序列的划分树(排序整数序列，调用过程 build(1, n, 0))。

2）反复输入查询(a, b, c)，通过调用函数 query(1, n, a, b, 0, c)计算和输出$[a, b]$区间中第 c 大的值。

程序清单

```
#include <stdio.h>
```

```
#include <iostream>
#include <string.h>
#include <algorithm>
using namespace std;
const int MAXN=100010;                          // 整数序列规模大小的上限
int tree[30][MAXN];                             // tree[p][i]表示第p层中第i位置的值
int sorted[MAXN];                               // 已经排序的数
int toleft[30][MAXN];                           // toleft[p][i]表示第p层从1到i中有多少
                                                // 个数被划入下一层的左子区间
```

构建划分树的过程build(l, r, deep)和查询函数query(L, R, l, r, deep, k)如前所述，这里不再赘述。

下面，分别给出输入整型数的外挂函数和主程序。

```
static inline int Rint()                        // Rint()为整型数输入函数。函数返回类型前
                                                // 加上关键字inline即把Rint()指定为内联
{
  struct X
  {
   int dig[256];                                // 标志有用码
   X()
   {
    for(int i='0';i<='9';++i) dig[i]=1;
    dig['-']=1;
   }
   };
   static  X fuck;
    int s=1, v=0,c;                             // 假定正数,整数值初始化为0
    for (; ! fuck.dig[c=getchar()];);           // 输入忽略串首的无用码
    if (c=='-') s=0;                            // 设负数标志
       else if (fuck.dig[c])v=c^48;             // 记录最高位的数值
    for(;fuck.dig[c=getchar()];v=v* 10+(c^48)); // 按照由高位至低位的顺序输入数码并计算整
                                                // 数值v
      return s ? v : -v;                        // 返回整数值
}

int main()                                      // 主程序
{
    int n,m;
    while(~scanf("%d %d",&n,&m))                // 反复输入数组的大小和要回答的问题数
    {
       for(int i=1;i<=n; i++)                   // 输入数组,排序数组初始化
        {
          scanf("%d",&tree[0][i]);
          sorted[i]=tree[0][i];
        }
      sort(sorted+1,sorted+n+1);                // 计算排序数组sorted[]
      build(1,n,0);                             // 从0层的根节点1出发,建立区间[1,n]的划分树
      while(m--)                                // 依次和处理m个询问
      {
       int a,b,c;
       scanf("%d %d %d",&a,&b,&c);              // 输入当前询问
       printf("%d\n", query(1,n,a,b,0,c));      // 计算和输出[1,n]的子区间[a,b]中第c大的值
      }
```

```
    }
    return 0;
}
```

在一些较复杂的综合题中，划分树经常被作为核心子程序，与其他算法配合使用。

【1.1.2 Super Mario】

【问题描述】

Mario 是世界著名的水管工。他的“魁梧”体格和惊人的弹跳能力经常让我们想起他。现在可怜的公主又有麻烦了，Mario 要去拯救他的情人。我们把到老板城堡路作为直线(其长度为 n)，在每个整数点 i、高度 h_i 的地方有一块砖。现在的问题是，如果 Mario 能跳的最大高度为 h，在 $[L, R]$区间内要有多少块砖，Mario 才可以踩踏上去。

输入：

第一行给出整数 T，表示测试用例的数目。

对每个测试用例：

第一行给出两个整数 n、m($1 \leqslant n \leqslant 10^5$，$1 \leqslant m \leqslant 10^5$)，$n$ 是路的长度，m 是查询次数。

下一行给出 n 个整数，表示每块砖的高度，范围是[0，1 000 000 000]。

接下来的 m 行，每行给出 3 个整数 L、R、h。($0 \leqslant L \leqslant R < n$，$0 \leqslant h \leqslant 1\,000\,000\,000$)。

输出：

对每个测试用例，输出“Case X:”(X 是从 1 开始的测试用例的编号)，然后给出 m 行，每行给出一个整数。第 i 个整数是针对第 i 个查询，Mario 可以踩踏上去的砖块的个数。

样例输入	样例输出
1 10 10 0 5 2 7 5 4 3 8 7 7 2 8 6 3 5 0 1 3 1 1 9 4 0 1 0 3 5 5 5 5 1 4 6 3 1 5 7 5 7 3	Case 1: 4 0 0 3 1 2 0 1 5 1

试题来源：2012 ACM/ICPC Asia Regional Hangzhou Online

在线测试地址：HDOJ 4417

试题解析

简述题意：查询区间$[L, R]$内不大于 H 的数字个数，即在 Mario 最大跳跃高度为 H 的情况下，他在$[L, R]$内可踩踏多少块砖。

本题是一道综合题，采用的算法是二分查找 + 划分树：若 Mario 能够跳过高度为 x 的砖头，则所有高度不大于 x 的砖头都能够跳过。这就凸显出问题的单调性，使得二分查找方法得以施展。

我们将区间$[s, t]$映射成大小值区间 = $[l, r]$ = $[1, (t-s)+1]$，每次计算中间指针mid =