《算法竞赛入门到进阶》：勘误和改进

出版社：清华大学出版社

作者：罗勇军 郭卫斌

最近更新：2019.8，第 3 次印刷

本书在多次重印过程中，进行了持续改进。本文记录了这些改进的细节，包括两方面的内容：

（1）勘误。本书的印刷或内容错误，每次新印刷时，会修改新发现的问题。

（2）新内容。增加的新内容。

本文下载地址：

（1）https://github.com/luoyongjun999/code

（2）QQ 群：567554289

请读者多提意见，非常感谢！联系方式：QQ 15512356，邮箱 15512356@qq.com

一、改进历史

时间	页码	改正内容
勘误：2019.7 第 2 次印刷已改	55 页	把(c, r)改成(r, c)
	294 页	把 k 改成 n
勘误：2019.8 第 3 印已改	23 页	“在排列问题中，如果要求输出所有的全排列”
	109 页	图 6.7(b)，3 改为 13
新内容：第 3 印		增加 32 个新视频
勘误：待改正	210 页	“ 总复杂度是 $O(n(\log_2 n)^2)$ ”
	254 页	把“直到所有边都在 T 中”改为“直到所有点都在 T 中”

二、勘误细节

1、第 2 次印刷已改正部分

（1）55 页，把(c, r)改成(r, c)

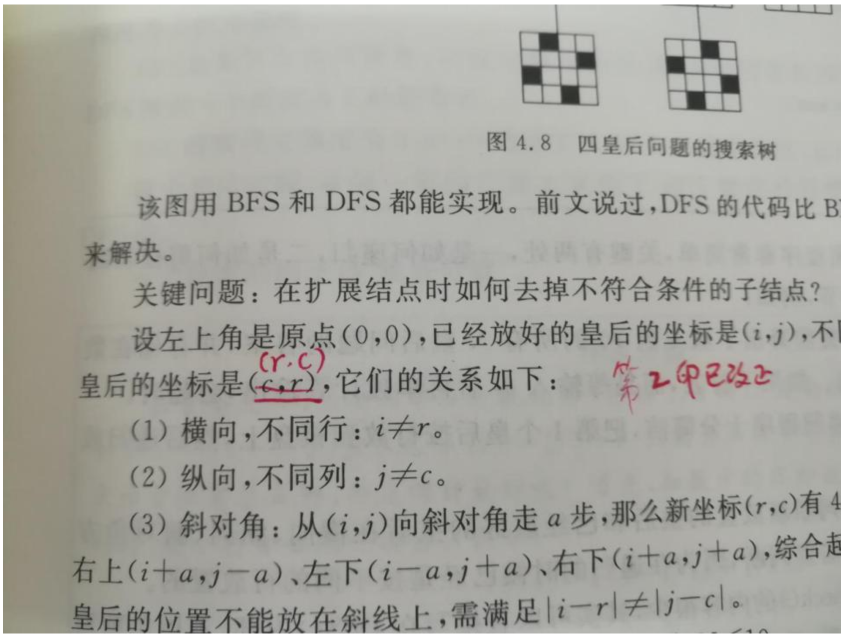


图 4.8 四皇后问题的搜索树

该图用 BFS 和 DFS 都能实现。前文说过，DFS 的代码比 B

来解决。

关键问题：在扩展结点时如何去掉不符合条件的子结点？

设左上角是原点(0,0)，已经放好的皇后的坐标是(i,j)，不

皇后的坐标是~~(c,r)~~ (r,c)，它们的关系如下：第2印已改正

(1) 横向，不同行：$i \neq r$。

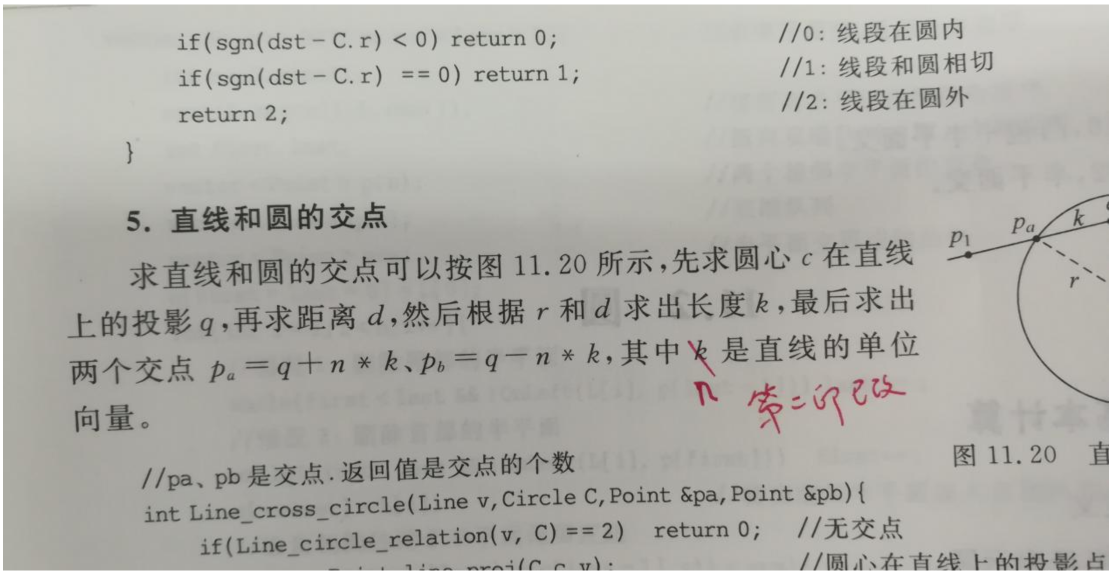
(2) 纵向，不同列：$j \neq c$。

(3) 斜对角：从(i,j)向斜对角走a步，那么新坐标(r,c)有4

右上$(i+a,j-a)$、左下$(i-a,j+a)$、右下$(i+a,j+a)$，综合起

皇后的位置不能放在斜线上，需满足$|i-r| \neq |j-c|$。

（2）294 页，把 k 改成 n。


```
    if(sgn(dst - C.r) < 0) return 0;          //0: 线段在圆内
    if(sgn(dst - C.r) == 0) return 1;         //1: 线段和圆相切
    return 2;                                 //2: 线段在圆外
}
```

5. 直线和圆的交点

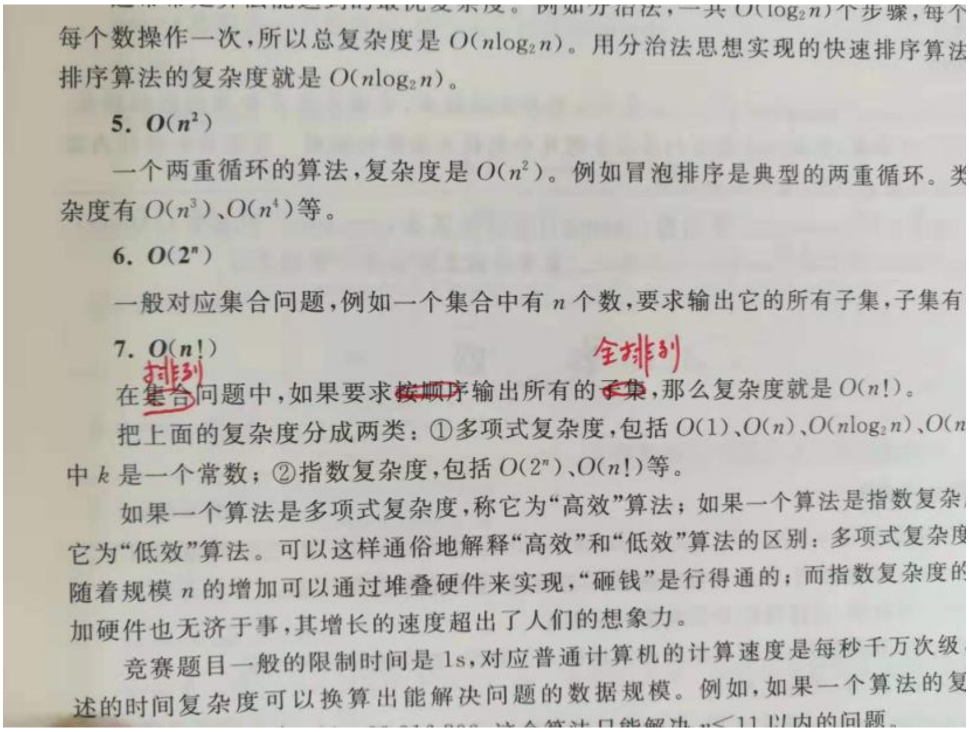
求直线和圆的交点可以按图 11.20 所示，先求圆心 c 在直线上的投影 q，再求距离 d，然后根据 r 和 d 求出长度 k，最后求出两个交点 $p_a = q + n * k$、$p_b = q - n * k$，其中 ~~k~~ n 是直线的单位向量。第二印已改

图 11.20 直

```
//pa、pb 是交点.返回值是交点的个数
int Line_cross_circle(Line v,Circle C,Point &pa,Point &pb){
    if(Line_circle_relation(v, C) == 2)   return 0;   //无交点
```

2、第 3 印已改正部分

（1）23 页，修改。


……例如分治法，一共 $O(\log_2 n)$ 个步骤，每个
每个数操作一次，所以总复杂度是 $O(n\log_2 n)$。用分治法思想实现的快速排序算法
排序算法的复杂度就是 $O(n\log_2 n)$。

5. $O(n^2)$

一个两重循环的算法，复杂度是 $O(n^2)$。例如冒泡排序是典型的两重循环。类
杂度有 $O(n^3)$、$O(n^4)$ 等。

6. $O(2^n)$

一般对应集合问题，例如一个集合中有 n 个数，要求输出它的所有子集，子集有

7. $O(n!)$

在~~集合~~排列问题中，如果要求~~按顺序~~输出所有的~~子集~~全排列，那么复杂度就是 $O(n!)$。

把上面的复杂度分成两类：①多项式复杂度，包括 $O(1)$、$O(n)$、$O(n\log_2 n)$、$O(n$
中 k 是一个常数；②指数复杂度，包括 $O(2^n)$、$O(n!)$ 等。

如果一个算法是多项式复杂度，称它为“高效”算法；如果一个算法是指数复杂
它为“低效”算法。可以这样通俗地解释“高效”和“低效”算法的区别：多项式复杂度
随着规模 n 的增加可以通过堆叠硬件来实现，“砸钱”是行得通的；而指数复杂度的
加硬件也无济于事，其增长的速度超出了人们的想象力。

竞赛题目一般的限制时间是 1s，对应普通计算机的计算速度是每秒千万次级
述的时间复杂度可以换算出能解决问题的数据规模。例如，如果一个算法的复

（3）109 页，应该是 13。

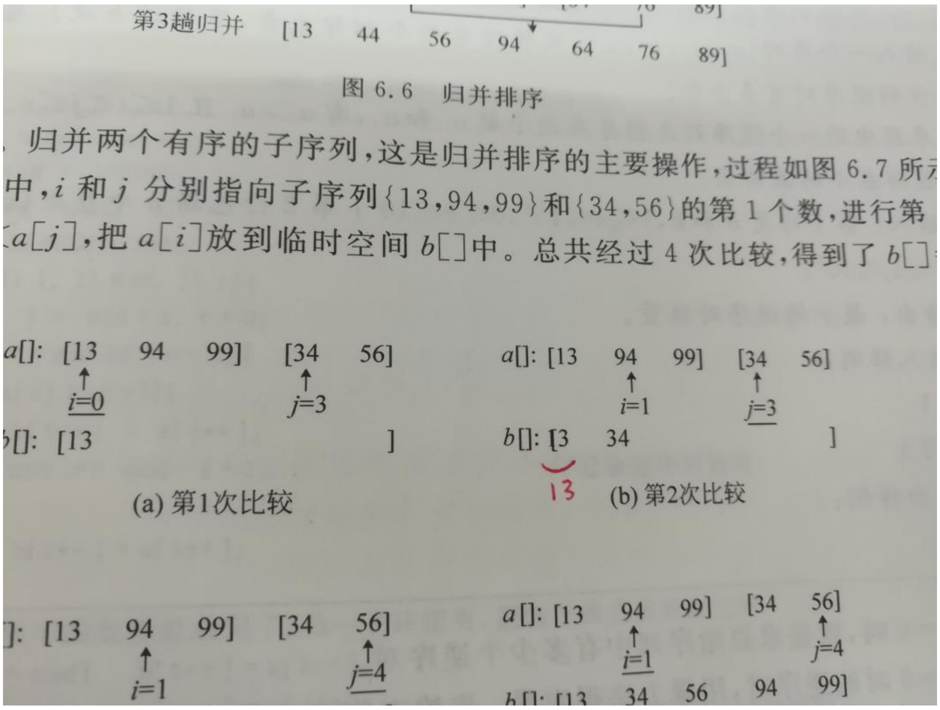
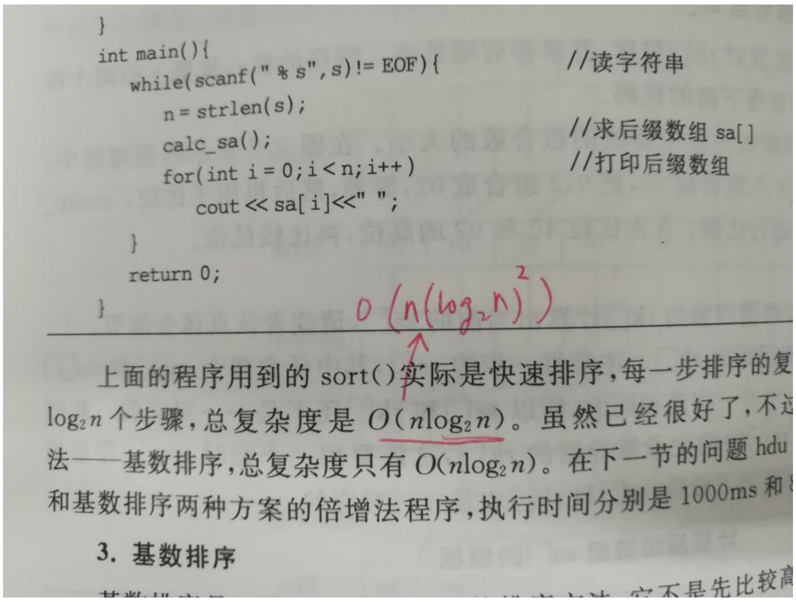

第3趟归并　[13　44　56　94　64　76　89]

图 6.6　归并排序

归并两个有序的子序列，这是归并排序的主要操作，过程如图 6.7 所示
中，i 和 j 分别指向子序列{13,94,99}和{34,56}的第 1 个数，进行第
$a[j]$，把 $a[i]$ 放到临时空间 $b[]$ 中。总共经过 4 次比较，得到了 $b[]$

3、待改正部分

（1）210 页：待改正。



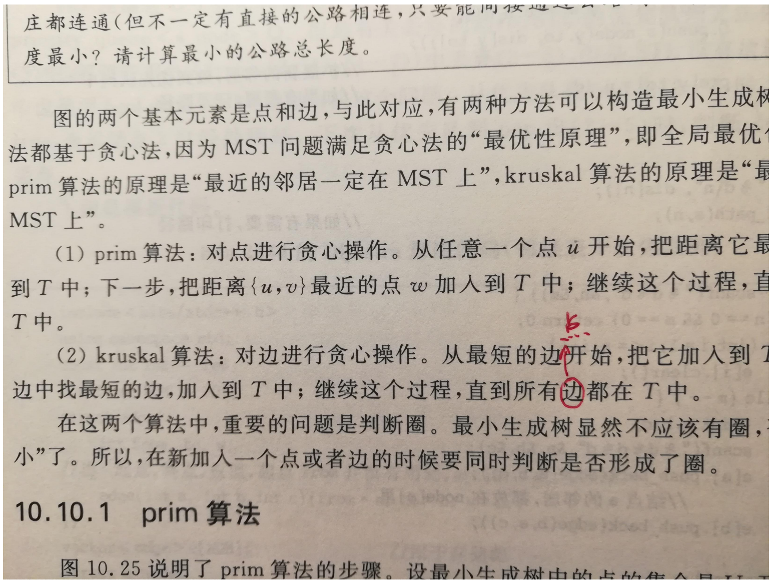
```
}
int main(){
    while(scanf("%s",s)!= EOF){          //读字符串
        n = strlen(s);
        calc_sa();                        //求后缀数组 sa[]
        for(int i = 0;i < n;i++)          //打印后缀数组
            cout << sa[i]<<" ";
    }
    return 0;
}
```

$O(n(\log_2 n)^2)$

上面的程序用到的 sort()实际是快速排序，每一步排序的复
$\log_2 n$ 个步骤，总复杂度是 $O(n\log_2 n)$。虽然已经很好了，不过
法——基数排序，总复杂度只有 $O(n\log_2 n)$。在下一节的问题 hdu
和基数排序两种方案的倍增法程序，执行时间分别是 1000ms 和

3. 基数排序

（2）254 页：“所有边”应该是“所有点”。待改正。



庄都连通(但不一定有直接的公路相连，只要能间接通过公路
度最小？请计算最小的公路总长度。

图的两个基本元素是点和边，与此对应，有两种方法可以构造最小生成树
法都基于贪心法，因为 MST 问题满足贪心法的“最优性原理”，即全局最优
prim 算法的原理是“最近的邻居一定在 MST 上”，kruskal 算法的原理是“最
MST 上”。

(1) prim 算法：对点进行贪心操作。从任意一个点 u 开始，把距离它最
到 T 中；下一步，把距离$\{u,v\}$最近的点 w 加入到 T 中；继续这个过程，直
T 中。

点

(2) kruskal 算法：对边进行贪心操作。从最短的边开始，把它加入到 T
边中找最短的边，加入到 T 中；继续这个过程，直到所有边都在 T 中。

在这两个算法中，重要的问题是判断圈。最小生成树显然不应该有圈，
小”了。所以，在新加入一个点或者边的时候要同时判断是否形成了圈。

10.10.1 prim 算法

图 10.25 说明了 prim 算法的步骤。设最小生成树中的点的集合是 U