

《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

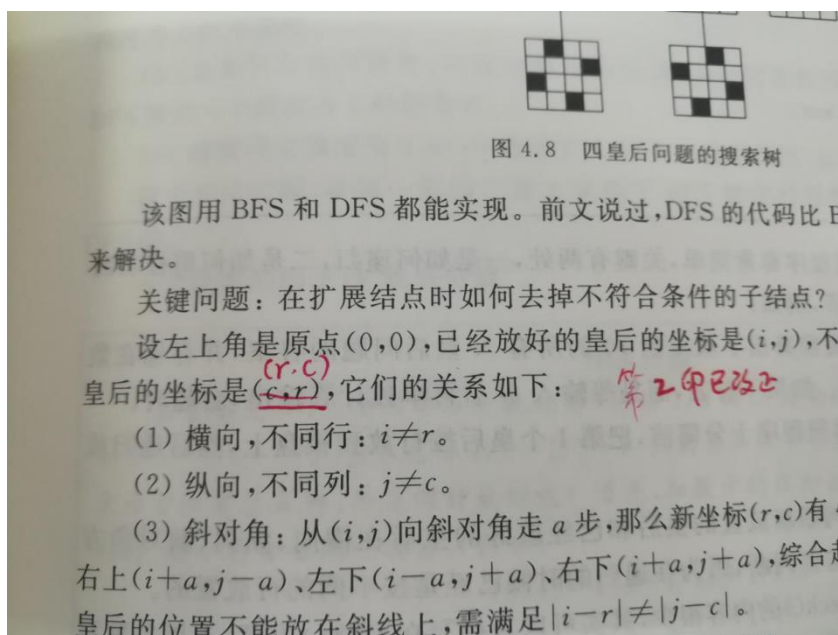
一、改进历史

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 $O(n(\log_2 n)^2)$ ”

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)

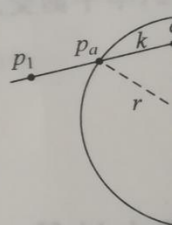


(2) 294 页, 把 k 改成 n 。

```
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 n 是直线的单位向量。



第 3 印已改

```
//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
    //圆心在直线上的投影点
```

图 11.20 直

2、第 3 印已改正部分

(1) 23 页, 修改。

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。复杂度有 $O(n^3)$ 、 $O(n^4)$ 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. $O(n!)$

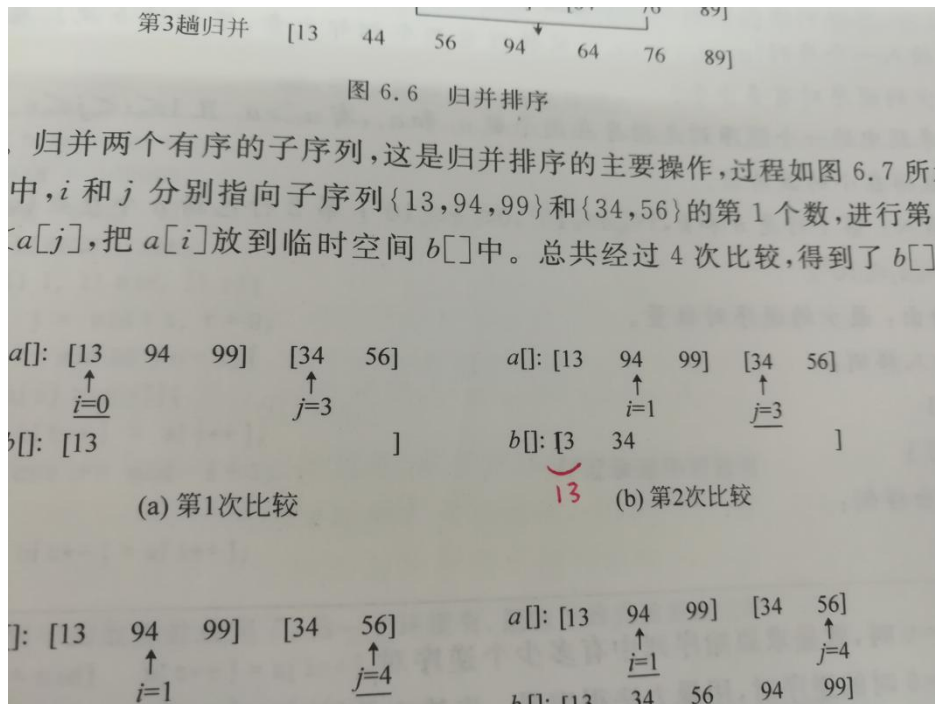
全排列
在集合问题中, 如果要求按顺序输出所有的子集, 那么复杂度就是 $O(n!)$ 。

把上面的复杂度分成两类: ①多项式复杂度, 包括 $O(1)$ 、 $O(n)$ 、 $O(n \log_2 n)$ 、 $O(n^k)$ 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度, 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。上述的时间复杂度可以换算出能解决问题的数据规模。例如, 如果一个算法的复杂度是 $O(n!)$, 那么只能解决 $n \leq 11$ 以内的问题。

(3) 109 页, 应该是 13。



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, 每一步排序的复杂度是 $\log_2 n$ 个步骤, 总复杂度是 $O(n \log_2 n)$ 。虽然已经很好了, 不过还有一种方法——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 100ms。

3. 基数排序

基数排序法不是先比较高