

《数据结构与算法》勘误表

URL: <http://db.pku.edu.cn/mzhang/ds/resource/errata.pdf>

张 铭

修定于 2006 年 12 月 4 日

目 录

一、教材信息.....	1
1. 教材出版信息.....	1
2. 每位作者负责的章节.....	1
3. 课程网站（课程讲义、算法源代码等）.....	1
4. 常见问题解答.....	1
5. 关于教材内容问题（包括错误和疑问）等数据结构技术性问题的讨论.....	1
6. 关于考研技巧问题，请到 BBS 考研版.....	1
二、关于纠错的说明.....	1
1. 本勘误表针对 2004 年 7 月第 1 次印刷版（V1），其他印刷版已经改正了某些错误。.....	1
2. 如果只是局部错误，则用红色标记出来。如果整体错误，则整体替换。.....	1
3. 第 77-83 中反复使用.size 和.str 这两个私有成员的问题建议基本修改为“strlen()”和“[]”。.....	1
三、勘误表（张铭编写）.....	2
四、对教材的一些解释.....	8
1. 关于栈的图示.....	8
2. 关于队列的图示.....	8
五、鸣谢.....	8

一、教材信息

1. 教材出版信息

(1) 主教材：许卓群、杨冬青、唐世渭、张铭，《数据结构与算法》，高等教育出版社，2004年7月。

(网上购书：<http://www.landrac.com.cn/book/bookdetail.asp?pluicode=14616-00>)

(2) 辅助教材：张铭、赵海燕、王腾蛟，《数据结构与算法——学习指导与习题解析》，高等教育出版社，2005年10月。ISBN 7-04-017829-X。辅助教材勘误表。

(网上购书：<http://www.landrac.com.cn/book/bookdetail.asp?pluicode=17829-00>)

(3) 主教材和辅助教材都在北大教材科、王府井图书大厦、西单图书大厦有售。某些新华书店也可能有售。

高教社购书热线：010-58581118, 58581117, 58581116。

高教社网上订购 URL：<http://www.landrac.com.cn>

2. 每位作者负责的章节

教材第 1、2、3 章（第 1-84 页）由许卓群主笔，第 4、5、6 章（第 85-202 页）由杨冬青主笔，第 8、10 章（第 254-293 页，第 333-358 页）由唐世渭主笔，第 7、9、11、12 章（第 203-253 页，第 294-332 页，第 359-468 页）由张铭主笔。本勘误表针对主教材。

辅助教材第 7、9、10、12、13、14 章（第 162-215、244-279、280-303、324-503 页）由张铭教授主笔，第 1、2、3、11 章（第 1-47、304-323 页）由赵海燕副教授主笔，第 4、5、6、8 章（第 48-161、226-243 页）由王腾蛟副教授主笔。

3. 课程网站（课程讲义、算法源代码等）

<http://www.db.pku.edu.cn/mzhang/DS/>

4. 常见问题解答

<http://db.pku.edu.cn/mzhang/ds/resource/FAQ.pdf>

5. 关于教材内容问题（包括错误和疑问）等数据结构技术性问题的讨论

请到 <http://db.pku.edu.cn/mzhang/DS/bbs/index.asp> 论坛注册账号，参与讨论。

请不要给张铭发 email 询问教材问题，该教材是四位作者分工合著。对于教材中所产生的概念或者排版错误（高教社用的是方正排版系统，因此图表公式都是出版社重新制作的，源代码格式也保持得不好），我们在此表示深深的歉意。

张铭非常忙，很可能没有时间直接回答您的问题。我们 BBS 上面有助教负责整理错误和问题，并且尽量解答，而且及时更新到勘误表和常见问题解答中。

6. 关于考研技巧问题，请到 BBS 考研版

请到 <http://bbs.pku.edu.cn/cgi-bin/bbstop?board=Kaoyan> 去询问。请不要给张铭发 email 询问考试重点、范围等问题。张铭要说的话，都体现在“2005 级硕士生入学考试数据结构复习大纲”

<http://db.pku.edu.cn/mzhang/ds/DSgradreview.HTM> 和

(http://db.cs.pku.edu.cn/mzhang/ds/account/a10_kydg.html) 中。

二、关于纠错的说明

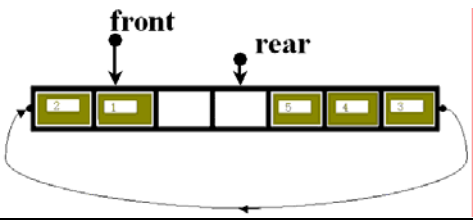
1. 本勘误表针对 2004 年 7 月第 1 次印刷版 (V1)，其他印刷版已经改正了某些错误。

2. 如果只是局部错误，则用红色标记出来。如果整体错误，则整体替换。

3. 第 77-83 中反复使用.size 和.str 这两个私有成员的问题建议基本修改为“.strlen()”和“[]”。

三、勘误表（张铭编写）

页数	具体位置	修改结果
P5	L7	C++中 二维数组的存储方法是 按照行优先顺序 ，每行以 $j=0,1,2,\dots,k-1$ (k 为列数) 的顺序存储其数组元素。
P9	倒数 L9	};
P10	L17	};
P12	倒数 L8	void matrix_addition(double **M1, double **M2, int k) {
P13	倒数 L6	$\sum_{i=1}^n (a \times (n - i + 1)) = (a \times (n \times (n + 1) / 2))$
P14	L15	double abs_biggest(double **M, int k)
P14	L21-24 增加一个“}”	current_big = temp; } // current_big 存储当前最大者
P16	L20	$rate T(n) = \Theta(G(n))$ <small>$n \rightarrow \infty$</small>
P21	L20	};
P22	倒数 L9	odelist[0],odelist[1],...,odelist[k-1]
P23	L3, 增加模版说明	template <class ELEM> class list {
P24	L2	#include <assert.h>
P24	L6	assert(curr_size<size)&&(curr>=0)&&(curr<=curr_len)
P24	L16 公式	$\sum_{i=0}^{k-1} p \times (k - i) \approx \frac{k}{2}$
P26	图 2.3 下面的定义	LisPtr first, last ;
P27	L4	// 指向第 i (i=0, 1, ..., n-1) 个结点，当链表中结点数 小于等于 i 时返回 1 带格式的：突出显示
P27	倒数 L3-1	void RemoveAfter(ListNode * link) { ListNode *newlink=link; if (newlink!=NULL){ ln->link = newlink->link; delete newlink; } if(ln->link == NULL) //处理 ln 指向尾结点的情况 Last = ln }
P29	倒数 L10	DbllListNode *first, *last;
P32	2.5 节第 1 行	(Last-In First-Out , LIFO)
P32	倒数 L2	Public: Stack(int s); ~Stack();
P33	添加 ClearStack(), 类定义添加分号“;”	void ClearStack() // { top = 0; } 清空栈内容 };
P33	倒数 L10 后半部分	最先压入的元素编号为 1 ，然后依次为 2,3,4 。

P35	倒数 L15-L11	Class Stack { //链式栈，假定其元素类型为 ListNode private: ListNode *top; public: Stack() //创建一个空栈，不用指定最大长度
P35	倒数 L4	void Stack::Push(float item)
P38-39	从倒数 L6 开始算法整体替换	(1) 当输入的是操作数时，直接输出到后缀表达式 PostfixExp 序列。 (2) 当输入的是开括号时，把它压栈。 (3) 当输入的是闭括号时，先判断栈是否为空，若为空（括号不匹配），应当进行错误异常处理，清栈退出。若非空，则把栈中的元素依次弹出，直到遇到第一个开括号为止，将弹出的元素输出到后缀表达式 PostfixExp 的序列中（弹出的开括号不放到序列中），若没有遇到开括号，说明括号也不匹配，进行异常处理，清栈退出。 (4) 当输入的是运算符 op（四则运算“+、-、*、/”之一）时 (a) 循环，当（栈非空 and 栈顶不是开括号 and 栈顶运算符的优先级不低于输入的运算符的优先级）时，反复操作：将栈顶元素弹出，放到后缀表达式序列中； (b) 把输入的运算符 op 压栈。 (5) 最后，当中缀表达式 InfixExp 的符号序列全部读入时，若栈内仍有元素，把它们全部依次弹出，都放到后缀表达式 PostfixExp 序列尾部。若弹出的元素遇到开括号时，则说明括号不匹配，进行错误异常处理，清栈退出。
P39	倒数 L3	Calculator(void) {};
P40	页首	添加一行类定义结束标记：};
P40-41	L8,L14,P41 倒数 L17	S.StackEmpty()改为 S.IsEmpty()
P40	Compute() 中 L4	double operand1,operand2;
P41	Run()算法中 L6	switch(c) {
P41	Run()算法倒数 L2	S.top()改为 S.GetTop()
P48	倒数两行	它在“加入”新元素时，限于表的一端进行(队列的尾端，称为“队列尾”)，而元素的“取出”则被限制于表的另一端(队列的前端，称为“队列头”)。
P49	倒数 L10	public:
P49	倒数 L1	};
P49	倒数 L7-L5	void EnQueue(T item); // item 进入队列尾端 T DeQueue(); //返回队列前端的元素内容，并从队列删去 T GetFirst(); //返回队列前端的元素内容，但不从队列删去
P50	图 2.17 及图上一行	图上一行中“图 12.7”改为“图 2.17”，图改为： 
P50	倒数 L1	};

P53	L24	public: Queue() //创建一个空队列, 不用指定最大长度 { front = rear = NULL; curr_len = 0; }; ... };
P59	L3	char s3[10]="";
P59	倒数 L5-L4	S3[20]改为 S3[30] int result;
P51	L6	assert(Qlist!=NULL);
P60	倒数 L9-L8	strchr(s,'r'); 结果返回指向第 11 个字符的指针。倒着寻找'r', strchr(s,'r'); 结果返回指向第 29 个字符的指针。
P72	倒数 L2-1	
P62	L22	String(char* s); 修改为 String(const String& s);
P64	L4	S.strlen()
P65	L4-6	“ If” 修改为小写“if”
P65	L15	friend ostream & operator <<(ostream & istr,String & s);
P68	倒数 L5	String s2 = s3="set";
P71	算法 3.2 中 L4	while(s[i]!='\0' && d[i]!='\0')
P72	算法 3.5 中 L6-L8	while(i >= 0 && d[i] !=ch) //循环跳过那些不是 ch 的字符 i--; if (i<0) //当本串不含字符 ch 时, 则在串尾结束
P72	算法 3.4 中的 L5,算法 3.5 中的 L5	int i = 0;
P73	代码 3.6 L7 代码 3.6 L11	private : };
P74	L6	// delete [] str;
P77	第 77-83 页算法	#include <string.h>改为 #include "String.h" 所有 “.size” 改为 “.strlen()”, “.str[]” 改为 “[]”
P77	算法 3.13 中 L8-L9	for (int g= startindex; g <= LastIndex; g++) {
P78	算法 3.14 中 L8,L9	加 “int” 表示 i,j 初次定义
P78	倒数 L6-L5	if (j >= P.strlen()) return(i-j);
P81	P81 倒 L1, P82L1	$n_4=k=3$, 由于 $q_7!=q_3$, 于是令 k'' 为 $n_{k'-1}$, $k'' = 1$ 。
P82	第一段最后两行	k' 为 $n_{k'-1}=1$, k'' 为 $n_{k'-1}=0$ 。循环结束时, $q_{11}=q_0$, $n_{11}=k'' + 1=1$ 。
P83	L7	for(i=startindex;i <S.strlen();i++)
P83	L15	if(j=P.size)改为 if (j==P.strlen())
P97	倒数 L9	using std::queue; //使用 STL 的队列
P99	倒数 L14, 搞反了	return((root)?false:true);
P99	P99 倒数 L10, P100 的 L5、倒数 L20	函数头后面的分号去掉
P100	L17	“ If” 修改为小写“if”

P100	倒数 14 行	if((temp==NULL) current==temp->rightchild()) // 出版社排版时, 给弄花了。
P101	L3	{ // 以后序周游的方式删除二叉树
P101	L10	去掉“}”后的分号
P104	图 4.12	D 的右指针添加指向 A 的虚线
P106	L8	if(root->leftchild() == NULL)
P106	L11	if (pre) root->lTag=1;
P113	L10, L18	if (pointer->leftchild() == NULL)
P113	L18	if (pointer->rightchild() == NULL)
P116-122	P116 倒 L3, P117 倒 L15, P118 L1,P122 L3	$\lfloor \frac{n-1}{2} \rfloor$ 修改为 $\lfloor \frac{n}{2} \rfloor - 1$
P118	图 4.22(d)部分错误	图(d)中 23 的左子结点值为 68
P118	倒数 L9-8	设有关键码集合 $\{K_0, K_1, \dots, K_{n-1}\}$, 若对 $i= u+1, u+2, \dots, n-1$ 均已满足
P118	倒数 L5	代码 4-17 中的 SiftDown 函数将范围扩大到对 $i = \mathbf{u}$ 上式也成立。
P121	L7	return heapArray[CurrentSize];
P121	Remove 算法 L7	heapArray[pos]=heapArray[--CurrentSize]; // 最后一个有效元素置到当前
P122	自公式(4-5)开始	$D = \sum_{i=1}^{\log n} (i-1) \frac{n}{2^i} \quad (\text{公式 4-5})$ <p>设 $x = \sum_{i=1}^{\log n} (i-1) \frac{1}{2^i}$ (公式 4.6)</p> <p>则有 $D = x \cdot n$</p> <p>公式 4.6 两边同乘以 2, 得到</p> $2x = \sum_{i=1}^{\log n} (i-1) \frac{2}{2^i} = \sum_{i=1}^{\log n} \frac{i-1}{2^{i-1}} = \sum_{i=1}^{\log n - 1} \frac{i}{2^i} \quad (\text{公式 4.7})$ <p>公式 4.7 与公式 4.6 相减, 得到</p> $x = \sum_{i=1}^{\log n} \frac{1}{2^i} - \frac{\log n}{2^{\log n}} = (1 - \frac{1}{2^{\log n}}) - \frac{\log n}{n} = 1 - \frac{\log n + 1}{n} \quad (\text{公式 4.8})$
P125	L9,L10	RemoveMin () // 注意大小写
P153	倒数两段	log n 修改为 log* n, log65536 = 4 修改为 log*65536 = 4
P153	倒数一句	这是均摊分析方法的一个例子, 读者可以查阅有关算法分析的参考资料。
P167	图 6.3 下面第 5 行文字	删除原文中“且路径长度大于等于 3”。
P190	倒数 L8-L7	if(D[G.ToVertex(e)].length>(D[v].length+G.Weight(e))) // &&G.Mark[G.ToVertex(e)]== UNVISITED 冗余
P191	图 6.17,P192Path 矩阵	顶点的编号都减 1, 以符合 C/C++ 编号习惯
P193	L5	D[i][j].pre=-1; //将路径设为-1 (-1 不是图的结点)
P193	算法 6.11 倒数 L3	D[i][j].pre=v; 改为 D[i][j].pre= D[v][j].pre;
P205	代码 7.2 中增加两个函数 le()和 ge()	static bool le(int x,int y) {return x<=y;} static bool ge(int x,int y) {return x>=y;}
P209	算法 7.6 倒数 L6	Array[j+1] = array[j]; // 将大于记录 i 的数据后移

P214	算法 7.10 倒数 L6	for (int j=n-1; j>i; j--) // 从 后往前 扫描剩余记录 // 关于修改的解释: 原算法也正确, 修改后跟 P214 倒数第 1 段的解释一致
P219	Shell 排序总时间代价公式	$\Theta(\sum_{i=0}^{\log n-1} (n^2 / 2^i)) = \Theta(n^2 \sum_{i=0}^{\log n-1} (1/2^i)) = \Theta(n^2 \cdot 2) = \Theta(n^2)$
P229	倒数 L13	if (Compare: : le (TempArray[index1], TempArray[index2]))
P230	L12	算法 7.10 -> 算法 7.14
P231-2 32	【算法 7.15】优化的两路归并排序, 除 ImprovedTwoWayMergeSorter 外整体替换	把 P230 倒数 L1 和 P231 L9-10 的 DoSort 修改为 Sort。删除原来那个 7 行的 Sort 函数, 删除算法 7.15 第 7 行的 DoSort 定义。 在 if (right-left+1 > THRESHOLD) { ... } // 保持原来 then 部分, 增加下面 else 部分 else { //如果序列长度小于等于阈值, 采用直接插入排序 ImprovedInsertSorter<Record,Compare> insert_sorter; Insert_sorter.Sort(&Array[left],right-left+1); }
P231	倒数 L11	if (Compare: : le (TempArray[index1], TempArray[index2]))
P232	倒数 L8 大小写错误	函数 BuildHeap 和 RemoveMax 的具体实现请参见前面 4.7 小节。
P233	L7 大小写错误	maxheap. RemoveMax (record);
P241	倒数 L10	参数 j -> 参数 i //关于修改的解释: 这个参数本身多余, 并不影响算法功能
P242	倒数 L5	即 d 在 $\Omega(n \log n)$ 中
P243	图 7.11 (b)	queue[8] -> 88 -> 58
P245	桶式排序辅助空间	$\Theta(n+m)$
P247	第四段 L1, L2	已经知道所有基于比较的排序算法的上限为 $O(n \log n)$, 那么它的下限呢? 能不能降到 $\Omega(n)$? 然后, 后面我们将失望地看到: 基于比较的排序算法的下限也为 $\Omega(n \log n)$
P249	L2	已经知道所有排序算法都需要 $O(n \log n)$ 的运行时间, 因此可以推导出排序问题需要 $\Theta(n \cdot \log n)$ 的运行时间。
P250	习题 7.18	本章中算法 7.14 给出的归并排序是通过递归来实现的
P261	自倒数 L8-L7 替换为右边文字 (增加计算“实际读到一簇数据需要的时间”)	继续做一下计算。其实, 定位到簇后, 读这一簇中数据的时间只有: [3(交错因子)×8(扇区/簇)÷256(扇区/道)×11.1 ms/圈] ≈ 1.04 ms 而加上平均寻道时间和旋转延迟时间, 实际读到一簇数据需要的时间是: [9.5 ms(平均寻道时间)] + [0.5(半圈旋转延迟)×11.1 ms/圈] + [3(交错因子)×8(扇区/簇)÷256(扇区/道)×11.1ms/圈] = { 9.5 + 5.55 + 1.04 }ms = 16.09ms
P293	习题 8.13 后半部分	采用 外部结点数为 8 的败者树方法对该文件生成初始顺串。请画出 最初产生的 败者树和 最终输出的 初始顺串。
P311	L11 公式	$210485_{13} = 2 \times 13^5 + 1 \times 13^4 + 0 \times 13^3 + 4 \times 13^2 + 8 \times 13 + 5 = 771932_{10}$
P312	L4	int ELFhash (char *key) {
P312	L7	h = (h << 4) + *key++;
P319	L4 添加文字修饰	若 M 是素数, $h_1(K) = K \bmod M$, 则可以定义 $h_2(K) = K \bmod (M-2) + 1$, 或者 $h_2(K) = [K / M] \bmod (M-2) + 1$ 。若 M 是任意数, $h_1(K) = K \bmod p$ (p 是小于 M 的最大素数), 可以定义 $h_2(K) = K \bmod q + 1$ (q 是小于 p 的最大素数)。 请注意, 尽管最后一个方案的并不能保障 $h_2(K)$ 与 M 互素, 但此方案还是经常被采用。

P327	习题 9.4 第二行	[1,200]改为 [1, 20 000]
P344	L2	(5) 有 k 个子结点的 非叶结点 恰好包含 k-1 个关键词。
P344	L9	所有的叶结点都在 第二层 (根为第 0 层) ，在每个结点里关键词是按字典顺序排列的。
P348	倒数 L1 文字修饰	如果 B 树的阶为 3，则称为“2-3 树”。其实，B 树只是 2-3 树的一种推广。
P355	L7	设 B 树包含 N 个关键词，因此有 N+1 个 扩展的外部结点 ， 这些外部结点 都在第 I 层。
P360	第一段图中第二行第一列下标错误	$\begin{pmatrix} a_{0,0} & a_{0,1} \dots & a_{0,n-2} & a_{0,n-1} \\ a_{1,0} & a_{1,1} \dots & a_{1,n-2} & a_{1,n-1} \\ \dots & \dots & \dots & \dots \\ a_{m-1,0} & a_{m-1,1} \dots & a_{m-1,n-2} & a_{m-1,n-1} \end{pmatrix}$
P360	第二段倒数 L3	另外，边界上的结点 $a_{0,j}(j=1, \dots, n-1)$, $a_{i,0}(i=1, \dots, m-1)$ 只有一个 前驱 结点； $a_{m-1,j}(j=0, \dots, n-2)$ 和 $a_{i,n-1}(i=0, \dots, m-2)$ 只有一个 后继 结点。
P368	L1	colnum 修改为 colhead
P373	图 11.8 括号表示法错误	(L1: (a, b), (L1, c, L2: (d)), (L2, e, L3: (f, g)), L3) ((a, b), ((a,b), c, (d)), ((d), e, (f, g)), (f, g))
P384	倒数 L8-L6 (3 行) 替换为 (右边两行)	if (pmax + K > S) return ::new LinkNode;
P402	倒数 L7	$\frac{P_i}{W}$ 是检索第 i 个内部结点所代表的关键词的概率， $\frac{q_i}{W}$ 是被检索的关键词属于第 i 个外部结点代表的可能关键词集合的概率。
P403	L2	计算 ASL(n)公式的第二行 li 前多一个左括号
P407	倒数 L3	“最右子结构”应为“最优子结构”
P408	图 12.13 第三步更正为	<p>第三步</p> <p>花费 51 43 52 41 40 49</p> <p>总权 24 24 24 22 22 24</p>
P411	L3	图 12.16(b)，结点 15 的左子树值为 12
P412	图 12.18	图(a) 中 a 结点平衡因子为+2，b 结点平衡因子为+1
P413	图 12.19 最下的图	图(a) 中 a 结点平衡因子为-1 或 0
P424	图 12.25	

P463	图 12.47(b)文字说明	(b) 连续两次访问 结点 T 时的伸展情况
------	----------------	-------------------------------

四、对教材的一些解释

1. 关于栈的图示

第 44 页图 2.11, 第 47-48 页图 2.15 这几个栈的图示, 都是基于所谓“下推表”的说法而画的。并不表示每次进栈、出栈时都需要移动栈中所有元素。我们认为, 一次进栈和一次出栈所花费的时间都只有 $O(1)$ 。

2. 关于队列的图示

第 49 页图 2.16 和第 50 页图 2.17 这两个队列的图示, 可以把队列所用的数组从右到左进行标号, 这样就跟第 51 页队列算法 2.13 和 2.14 中的下标操作一致。

第 50 页图 2.18 和第 52 页图 2.19 与通常情况下数组的下标表示方法一致, 是从左到右进行标号的。

五、鸣谢

感谢同时任教的赵海燕、王腾蛟老师、历届助教的大力协助, 全体信息学院同学的认真钻研、积极探索, 所有热心读者的帮助。