数据结构（第三版）教材勘误表

注：其中部分更正是为了更便于读者理解，并非原文有误

前言和第一章

页码	行	原内容	更正内容	备注
前言III	倒 15	由 N 个算法 $\mathbf{A}, \mathbf{A}^1, \mathbf{A}^2, \cdots, \mathbf{A}^N$ 组成	由 N 个算法 $\mathbf{A}^1, \mathbf{A}^2, \cdots, \mathbf{A}^N$ 组成	
2	倒 3	整数是包括负的、零和正的全体数	整数是如下负的、零和正的全体数	
10	倒 4	$\binom{r}{k}=\begin{cases}\dfrac{r(r-1)\cdots(r-k+1)}{k(k-1)\cdots(1)}=\dfrac{r^k}{k!}=\prod_{j=1}^{k}\dfrac{r+1-j}{j} & \text{整数 } k\geq 0\\ 0 & \text{整数 } k<0\end{cases}$	式(1-42)中两个等号中间的 $\dfrac{r^k}{k!}$ 去掉，以及去掉一个多余的等号	

第二章

页码	行	原内容	更正内容	备注
31	17	在第 8 章将详细介绍索引存储。	在 8.6 节将介绍索引存储。	
40	8	置 max ← min ← A[1]	max ← min ← A[1]	“置”字去掉
40	倒 1	置 i← 1	i ← 1	“置”字去掉
41	4	置 i ←i+1	i ← i+1	“置”字去掉
41	倒 2	置 mid ←	mid ←	“置”字去掉
42	4	置 fmax← max{gmax, hmax}	fmax← max{gmax, hmax}	“置”字去掉
42	5	置 fmin← min{gmin, hmin}	fmin← min{gmin, hmin}	“置”字去掉

第三章

页码	行	原内容	更正内容	备注
49	12	n 为自然数。当 $n=0$ 时，	。当 $n=0$ 时，	

49	20	确定线性表是否为空。	判断线性表是否为空。	
49	最后一行		加入“⑧归并、分拆、赋值、排序……”	
51	4	// 在顺序表 *A* 中下标	/*在顺序表 *A* 中下标	第 4、5 行都是注释
51	5	序表的当前长度	序表的当前长度*/	
54	8	$\frac{0+1+\cdots+n+n+1}{n+2}=\frac{1}{n+2}\frac{(n+1)(n+2)}{2}=\frac{n+1}{2}=O(n)$		
58	4	算法 DLInsert(*s*, *p*. head)	算法 DLInsert(*s*, *p*)	
58	6	right (*s*)←*p*.	right (*s*)←tail← *p*.	
58	倒 6	算法 DeleteNode(s. head)	算法 DeleteNode(s. head, tail)	
60	17	① push(item)	① push( )	
60	18	② pop(item)	② pop( )	
60	19	③ peek(item)	③ peek( )	
62	倒 6	算法 Clear(.top)	算法 Clear(top)	
63	倒 12,13 行	在 12,13 行之间加入一行	CREATE(S).	
63	倒 9	THEN Stack.push(string[i]).	THEN S ⇐ string[i].	
63	倒 6	THEN (IF(Stack.IsEmpty())	THEN (IF(IsEmpty (S))	
64	4,5 行中间		加 ‘)’	缺失一个右括号
64	7	IF (!Stack.IsEmpty())	IF (NOT IsEmpty(S))	
64	18	算法 Factorial(n)	算法 Factorial(n. m)	
64	21	IF *n* = 0 THEN (m←1. RETURN m .)	IF *n* = 0 THEN (m←1. RETURN.)	
65	倒 7	IF *n* = 0 THEN (m←1. RETURN m .)	IF *n* = 0 THEN m←1.	有返回值时，RETURN 语句省略
65	倒 6	ELSE (Factorial(*n*-1) . t). m←n*t.)	ELSE (Factorial(*n*-1 . t). m←n*t.)	
66	表 3.15	参数 long n	参数 n	
68	倒 8	算法 QDelete(A, item. A)	算法 QDelete(A . item, A)	
69	倒 13	算法 QInsert(item . front)	算法 QInsert(item . front, rear)	

第四章

页码	行	原内容	更正内容	备注

79	11	即非对角线上的元素	即非主对角线上的元素	
81	9	j=0.	j←0.	
81	17	val(b[j])	value(b[j])	
83	倒 7	稀疏矩阵的表示方式为正交链表	稀疏矩阵的表示方式为十字链表	
86	9	i=0.	i←0.	
86	12	p[i]=s[i].	p[i] ←s[i].	
86	14	p[i] ='\0'.	p[i] ←'\0'.	
87	10	IF *j* = | *P* | THEN (Position←*i* – | *P* | . RETURN Position.)	IF *j* = | *P* | THEN Position←*i* – | *P* | .	有返回值时，RETURN 语句省略
87	14	RETURN Position.	此语句去掉	同上
88	7	几个等号的位置	4 个竖等号与后面的 4 个字符对齐	
89	17	IF i<m THEN (Position← –1. RETURN Position.)	IF i<m THEN Position← –1.	同上
89	19	Position← j–m. RETURN Position.	Position← j–m.	同上
89	倒 7	倒 7 至倒 9 有两行半叙述错误	“再检验 $p_{h+1} = p_{j+1}$ 是否成立”之前的两行半是错误的，删掉。将“②”提前到倒数 10 行开头	
92	倒 15	存放该数组至少需要多少个字节	存放该数组需要多少个字节	

第五章

页码	行	原内容	更正内容	备注
94	倒 9	一棵树是结点的有限集合 T。	一棵树是结点的有限集合 *T*。*T* 空时为空树。	
99	倒 10	它或者是空集	它或者是空集，称为空二叉树	
111	5	即入栈、出栈和访问	即入队、出队和访问	
112	6	IF *ch* = *tostop* THEN (*t* ← Λ . RETURN *t* .)	IF *ch* = *tostop* THEN (*t* ← Λ . RETURN.)	
113	倒 12	IF *t* = Λ OR *p*= Λ THEN	IF *t* = Λ OR *p*= Λ OR *p*= *t* THEN	*p* 为根结点
117	7 和 8	*p*⇐*S*. *t* ←*p*.	*t*⇐*S*.	
120	倒 17	RETURN *q*.	此语句去掉	
120	倒 9	RETURN *q*.	此语句去掉	

121	9	RETURN *q*.	RETURN.	
121	13	RETURN *q*.	此语句去掉	
121	倒 7	RETURN *q*.	RETURN.	
121	倒 1	RETURN *q*.	此语句去掉	
125	9	In Thread(Left(r) , pre . pre).	InThread(Left(r) , pre . pre).	In Thread 中的空格去掉
125	倒 14	Right(pre) = head. RThread(pre) = 1 **.**	Right(pre) ← head**.** RThread(pre) ← 1.	另外两语句间加一空格
129	13,14		这两行应归入前面的③中	
129	倒 10	利用 *Left* 为空标识线索	利用 RThread 为空标识线索	
129	倒 9	利用 *Right* 为空标识线索	利用 LThread 为空标识线索	
130	倒 5	第一个结点的 *Left* 指针和最后一个结点的 *Right* 指针	第一个结点的 *Pred* 指针和最后一个结点的 *Succ* 指针	
134	倒 6	然后继续运行，生成外结点为$(w_1+w_2, w_3,\ldots,w_n)$的哈夫曼树$T^*$.	然后继续运行，生成 n 个外结点的哈夫曼树 T^*. 往证 T^*是最优二叉树。	
135	4	$E(T')=E(T'')$.	$E(T')=E(T'')+(w_1+w_2)$.	
135	5	一方面，由归纳假设，T^*是由以n-1个外结点$(w_1+w_2, w_3,\ldots,w_n)$构成的最优二叉树，则 $E(T^*) \leqslant E(T'') = E(T') \leqslant E(T)$.	一方面，设 T^{**}是由 n-1 个外结点$(w_1+w_2, w_3,\ldots,w_n)$生成的哈夫曼树，由归纳假设，$T^{**}$是由 n-1 个外结点$(w_1+w_2, w_3,\ldots,w_n)$构成的最优二叉树，则 $E(T^{**}) \leqslant E(T'')$，从而 $E(T^*) = E(T^{**}) + (w_1+w_2)$ $\leqslant E(T'') + (w_1+w_2) = E(T') \leqslant E(T)$	
143	倒 9	THEN RETURN $q \leftarrow$ *FistChild* (p) .	THEN ($q \leftarrow$ *FirstChild* (p). RETURN.)	
143	倒 7	RETURN $q \leftarrow \Lambda$.	$q \leftarrow \Lambda$.	
143	倒 2	THEN RETURN $q \leftarrow$ *NextBrother* (p).	THEN ($q \leftarrow$ *NextBrother* (p). RETURN.)	
144	1	RETRUN $q \leftarrow \Lambda$.	$q \leftarrow \Lambda$.	
144	倒 4	*GetFistChild* (*t* . *child*);	*GFC* (*t* . *child*);	
145	1	GetNextBrother (*child*. *child*).	GNB (*child*. *child*).	
145	10	则将其大兄弟结点压入栈，且将该兄弟结点设为结点 *p*	则将该兄弟结点设为结点 *p*	入栈是在步骤①作的
145	倒 7	$p \Leftarrow S$.	($p \Leftarrow S$.	缺左括号
147	11	IF $t = \Lambda$ OR $p = \Lambda$ THEN RETURN.	IF $t = \Lambda$ OR $p = \Lambda$ OR $p = t$ THEN RETURN.	
147	11-17	IF $t=\Lambda$ OR $p=\Lambda$ THEN RETURN.	IF $t=\Lambda$ OR $p=\Lambda$ OR $p=t$ THEN RETURN.	调整语句位置，14 行挪到 17 行。

		FF2. [找到 *t* 所指结点的第一棵子树] *Q*←*FirstChild* (*t*). IF *q*=*p* THEN (result ← t. RETURN.) . FF3.[从 *t* 的第一棵子树开始依次搜索各子树，若搜索到则返回] WHILE *q* ≠ Λ DO (FindFather(*q* , *p*. *result*)	FF2. [找到 *t* 所指结点的第一棵子树] *q*←*FirstChild* (*t*). FF3.[从 *t* 的第一棵子树开始依次搜索各子树，若搜索到则返回] WHILE *q* ≠ Λ DO (IF *q*=*p* THEN (result ← t. RETURN.) . FindFather(*q* , *p*. *result*)	
149	图 5.50		删掉第 3 棵树 图 5.50 先根序列为 ABCD 的 3 棵树	
151	倒 1	READ(*n*, *m*, *p*).	READ(*n*, *m*, *q*).	
152	7	FOR *i* ← 1 TO *q* DO (	FOR *i* =1 TO *q* DO (	
152	倒 9	让 *x* 所在的树的根结点指向 *y* 所在的树	让 *x* 所在树的根结点指向 *y* 所在树的根结点	
152	图 5.53	A C D 0 0 0	A C D 1 3 4	
155	13	*Father*[*x*]保存结点 *x* 的父亲地址。	*Father*[*x*]保存结点 *x* 的父亲的地址。	
155	倒 12	*Father*(*x*) ← 0.	*Father*[*x*] ← 0.	圆括号改方扩号，下同
155	倒 7	IF *Father*(*x*) ≤ 0	IF *Father*[*x*] ≤ 0	
155	倒 5	*fx*←FIND(*Father*(*x*)).	*fx*←FIND(*Father*[*x*]).	
155	倒 4	*Father*(*x*) ← *fx*.	*Father*[*x*] ← *fx*.	
156	7	*Father*(*fy*)←*fx*.	*Father*[*fy*]←*fx*.	

第六章

页码	行	原内容	更正内容	备注
166	脚注		弱连通图的概念：将有向图的所有边略去方向后，如所得到的无向图是连通图，则该有向图为弱连通图。	
168	图 6.5		(b)矩阵中最后一个元素应为 0	
171	8	当所有顶点均被访问，整个深度优先遍历过程结束。	当所有 v_0 可及的顶点均被访问完，整个深度优先遍历过程结束。	

172	11	说明所有图中所有顶点都已被访问一次，	说明图中所有 A 可及的顶点都已被访问一次，	
173	倒 17	直到连通图中的所有顶点全部访问完为止。	直到图中所有 v_0 可及的顶点均被访问完为止。	
174	倒 11	$(0 \le i \le j < k)$	$(0 \le j < k)$	
174	倒 6	要说明的是，以上的遍历都是针对连通图而言的。对于非连通图，从任一顶点出发，	要说明的是，以上的例子都是针对连通图而言的。对于非连通图，从某些顶点出发，	
175	倒 10	就是把 AOV 网中的所有顶点排成一个线性序列	就是把 AOV 网中的所有顶点排成的一个线性序列	
176	8	如果将已经输出的顶点及边从网中删除	如果将已经输出的顶点及其出边从网中删除	
177	12、14	count[i] = top ; top = i ; j = top ; top = count[top] ;	两个语句间加个空格	
178	4	// 弹出堆栈顶点 j	// 堆栈弹出顶点 j	
178	5	PRINT " j" .	PRINT (j) .	
178	16	知排序开始时	知拓扑排序开始时	
178	倒 4	除了执行次序的先后关系外	任务间除了执行次序的先后关系外	
179	倒 7	以顶点 v_i 为尾的弧表示活动 a_i 的	以顶点 v_i 为弧尾的活动 a_i 的	
180	5	首先要求得到 AOE 网	首先要求得 AOE 网	
180	11	$ve(j)=\max_i\{ve(i)+weight(<i,j>)\mid <i,j>\in E(G), j=2,\cdots,n\}$	$ve(j)=\max_i\{ve(i)+weight(<i,j>)\mid <i,j>\in E(G)\}, j=2,\cdots,n$	$j=2,\cdots,n$ 在大括号外面
180	11		$vl(j)=\min_k\{vl(k)-weight(<j,k>)\mid <j,k>\in E(G)\}, j=n-1,\cdots,1$	同上
180	倒 13	改为 $\begin{cases} ve(1)=0 \\ ve(j)=\max_i\{\ ve(i)+weight(<i\ ,\ j>)\ \mid <i,j>\in E(G)\ \}, j=2\ ,...,n \end{cases}$		

180	倒 4	改为 $\begin{cases} vl(n)=ve(n) \\ vl(j)=\min\limits_{k}\{\ vl(k)-weight(<j,k>)\mid <j,k>\in E(G)\ \},j=n-1,...,1 \end{cases}$		
182	2	PRINT "< i, k> is Critical Activity! "	PRINT "< ", i, ",", k, "> is Critical Activity! "	
182	6	最早开始时间 $e(k)$和最迟开始时间 $l(k)$	最早开始时间 e 和最迟开始时间 l	
183	倒 2	路径“成本”是指该路径的权值累加和。	路径“成本”是指路径上边的权值的累加和。	
184	倒 9	最短路径以及边 (v,w) 到达 w	最短路径以及边 (v,w) 或 $<v,w>$ 到达 w	
188	14	修改 u 邻接顶点的 s[]值、path[]值和 dist[]值	修改 u 邻接顶点的 path[]值和 dist[]值	
188	倒 1		去掉最后一句“若图顶点多而边少，则算法的时间复杂性可认为是 $O(n^2)$。”	
190	7, 10, 14, 21	序列号	序号	
190	20		$A^{(k)}[i][j]=\min\{A^{(k-1)}[i][j],A^{(k-1)}[i][k]+A^{(k-1)}[k][j]\},0\le k<n$	$0\le k<n$ 在大括号外
192	9	和 $p^{(1)},\ldots,p^{(i-1)},\ldots,p^{(k)}$ 的并集	和$\{p^{(1)},\ldots,p^{(i-1)},p^{(i+1)}\ldots,p^{(k)}\}$的并集	
193	倒 1	令(p,C)是 Q 所有元素中	(令(p, C)是 Q 所有元素中	加左括号
194	1	并从 Q 中删除 p .	并从 Q 中删除(p, C) .	
194	3	将这些最短路径以及其约束	将这些最短路径及其约束	
194	3	添加到 Q 中.	添加到 Q 中.)	加右括号
200	表 6.4	第 2 行，第 3 列的①和③显得太大	①和③改小一些	
201	20	IF (Find(Vex1)≠Find(Vex2)) THEN	IF Find(Vex1)≠Find(Vex2) THEN	
201	倒 10	j←j+1. //扫描下一条边) ▌	j←j+1.) ▌ //扫描下一条边	
201	倒 5	为减少查询次数	为减少查询操作的复杂性	
202	倒 5	各个互不相交树 $T_i=(V_i,E_i)$ 的顶点集合	互不相交树 $T_i=(V_i,E_i)$ 的集合	
204	13	i≠i, < V_i， V_j >∉ E	i≠j, < V_i， V_j >∉ E	
204	倒 2	$WSM^{(0)}$=A	$WSM^{(0)}$=A（加上主对角线元素为 1）	

205	倒 11	p←Head[*i*].	p ← adjacent(Head[i]).	
205	倒 9	j←Vertex(p).	j ← VerAdj(p).	

第七章

页码	行	原内容	更正内容	备注
218	4	如果 $K_a < K_b$, $K_a = K_b$, $K_b < K_a$ 等三种可能性中	在 $K_a < K_b$, $K_a = K_b$, $K_b < K_a$ 三种可能性中	
218	6	条件①和②	性质①和②	
218	6	任何满足①和②的关系“<”都可以按本章	任何具有满足①和②的关系“<”的记录都可以按本章	
220	倒 11	需做 *j* 次关键词比较， *j*+1 次记录移动	需作 *j* 次关键词比较和 *j*+1 次记录移动	
221	表 7.1	平均情况 (*n*-1)(*n*+4)/2	(*n*-1)(*n*+4)/4	
222	11	注意这种排序仅限于组内关键词易位	注意这时排序仅限于组内关键词易位	
222	图 7.2	(b) (c) (d)	(b)的记录为 503 087 154 061 612 170 765 275 653 426 512 509 908 677 897 703 (c)的记录为 503 087 154 061 612 170 512 275 653 426 765 509 908 677 897 703 (d)的记录为 154 061 503 087 512 170 612 275 653 426 765 509 897 677 908 703	
223	9	FOR *j* = *d* TO *n* DO	FOR *j* = *d*+1 TO *n* DO	
225	倒 9	做 *n*–1 次关键词比较	作 *n*–1 次关键词比较	
225	倒 7	做 *n-i* 次关键词比较	作 *n-i* 次关键词比较	
226	图 7.4		第 3 趟的终止位置线应在 10 和 11 之间	
226	倒 14	这两种排序算法有一个共同特点，即对每个元素来说，只有当排序结束时，它才能进入正确的排序位置。	删去此句。	
227	倒 4	关键词都小于等于 *R*	关键词都小于等于 *R* 的关键词	
227	倒 3	关键词都大于 *R*	关键词都大于 *R* 的关键词	
228	16	RETURN *j*.	此语句去掉	
228	21	*j* ← Partition (*R*, *m*, *n*) .	Partition (*R*, *m*, *n*. *j*) .	
228	倒 6	是当指针 *i* 和 *j* 相遇或交叉时	是当指针 *i* 和 *j* 交叉时	

234	表 7.3	关键比较次数	关键词比较次数	
235	图 7.9	第二列，k5 和 k6 的比较结果为 94，漏掉了	在第二列加上 94	
237	13	ELSE *j*←*e*.) //终止循环 ▌	ELSE *j*←*e*.) ▌ //终止循环	
237	19	Restore(R, i, n)	Restore(R, i, n. R)	
237	22	Restore(R, 1, i-1)	Restore(R, 1, i-1. R)	
239	倒 2	$O((n+e)\log_2 n)$	$O(e+n\log_2 n)$	
240	9	*t*、*m* 是待合并文件 1 的头、尾指针，*n* 是待合并文件 2 的尾指针	*t*、*m* 是待合并文件 1 的起止位置，*n* 是待合并文件 2 的结束位置	
241	倒 7	剩余部分长度< *L*	剩余部分长度≤*L*	
244	表 7.5	表中第一行第一个数据 *O*(*n*log*n*)	*O*(*n*log$_2$*n*)	
244	倒 3	对于以元素比较为基础	对于以关键词比较为基础	
245	5	对 *X*(*p*), …, *X*(*q*) 进行排序	对 X_p, …, X_q 进行排序	
245	8	RETURN(InsertSort (*p*, *q*, *X*)).	(InsertSortA (*X*. *p*, *q*. *X*). RETURN.)	
245	10	*m* ←DIVIDE(*p*, *q*)	DIVIDE(*X*. *p*, *q*. *m*) .	
245	11	X_1←DCSort (*p*, *m*, *X*). X_2←DCSort (*m*+1, *q*, *X*).	DCSort (*p*, *m*, *X*. X^1). DCSort (*m*+1, *q*, *X*. X^2).	
245	12	对文件 *X*(*p*) , … , *X*(*m*)) 和 *X*(*m*+1) , … , *X*(*q*))排序	对文件（X_p , … , X_m) 和（X_{m+1} , … , X_q)排序	
245	13	RETURN COMBINE(X_1, X_2). //组合两个排序结果 X_1 和 X_2 ▌	COMBINE(X^1, X^2 . *X*). ▌ //组合两个排序结果 X^1 和 X^2	
246	倒 3	比较次数皆大于等于 $\Omega(n\log_2 n)$	比较次数皆大于等于 $n\log_2 n$	
247	倒 3	分“堆”或曰“分桶”	分“堆”或曰分“桶”	
248	倒 3	*r* 个桶为队列 Q_0, Q_1, … , Q_{r-1}	*r* 个桶为队列 Q_0, Q_1, … , Q_{r-1}	
248	倒 3	合并 Q_0, Q_1, … ,	合并 Q_0, Q_1, … ,	
249	倒 10	*m* 为基数	*r* 为基数	
249	倒 6	创建队列 Q_0, Q_1, …, Q_{m-1}	创建队列 Q_0, Q_1, …, Q_{r-1}	
249	倒 3	置队列 Q_0, Q_1, …, Q_{m-1} 皆为空	置队列 Q_0, Q_1, …, Q_{r-1} 皆为空	

249	倒 1	$X \leftarrow$KEY(R). $K_j \leftarrow X$ 的第 j 位.	这两个语句间加空格	
250	1	KEY(R) = $X = X_p K_{p-1} \ldots K_1$	KEY(R) = $X = K_p K_{p-1} \ldots K_1$	
250	7	FOR k = 0 TO m-1 DO　//依序将队列 Q_0, Q_1, …, Q_{m-1}	FOR k = 0 TO r-1 DO　//依序将队列 Q_0, Q_1, …, Q_{r-1}	
250	倒 9		这一段是对整个内排序的总结，应该与前面隔开些距离	
267	习题 7-9	冒泡排序算法关键词交换的次数	冒泡排序算法记录交换的次数	
267	习题 7-10	如果经过一趟冒泡和一次下沉后发现 R_j 和 R_{j+1}（1≤j≤n-1）没有交换	如果经过一趟冒泡和一次下沉后发现 R_j 和 R_{j+1}（1≤j≤n-1）相互没有交换	
267	18	试证明经过一趟起泡后	试证明经过一趟冒泡后	
268	16	WHILE　②　DO	WHILE　②　DO $j \leftarrow j-1$.	
268	习题 7-25	改写快速排序算法	改写快速排序的递归算法	
268	倒 1	堆中任意结点的关键词均小于它的左儿子和右儿子的关键词	堆中任意结点的关键词均小于等于它的左儿子和右儿子的关键词	

第八章

页码	行	原内容	更正内容	备注
271	10	1962 年，俄国数学家	1962 年，前苏联数学家	
272	16	树结构查找包括“二叉查找树”、“最优二叉查找树”、“高度平衡树”和“B 树”等查找方法。	引号都去掉	
273	1	置 $i \leftarrow$ l .	置 $i \leftarrow 1$.	
279	10	它的变化率 δ	它的变化量 δ	
279	倒 10	(置 $i \leftarrow \lceil$N/2$\rceil$. $m \leftarrow \lfloor N/2 \rfloor$.)	置 $i \leftarrow \lceil$N/2$\rceil$. $m \leftarrow \lfloor N/2 \rfloor$.	括号去掉
281	12	否则 (**$i \leftarrow i$** - DELTA[j]. $j \leftarrow j + 1$. 转 **C2**).	否则 **$i \leftarrow i$** - DELTA[j]. $j \leftarrow j + 1$. 转 **C2.**	括号去掉
281	14	否则 (**$i \leftarrow i$** + DELTA[j]. $j \leftarrow j + 1$. 转 **C2**).	否则 **$i \leftarrow i$** + DELTA[j]. $j \leftarrow j + 1$. 转 **C2.**	括号去掉
283	2	任一给定 k 的斐波那契树	任一给定 k 的斐波那契树 T_k	
283	8	例如，3 = 5 - f_3	例如图 8.5 中，3 = 5 - f_3	

287	倒 12	在字符次序上	在字典序上	
287	倒 1	若查找失败，则将用 K 作关键词的新记录插入 $\boldsymbol{T_B}$.	此句重复，删除	
288	11	//本算法成功结束	//本算法插入新结点后结束	
289	11	$q \leftarrow r$	$t \leftarrow r$	与后面算法保持一致
289	19	$q \leftarrow s$（s 取代 q）	$t \leftarrow s$	与后面算法保持一致
290	6	算法 D (q, a . q)	算法 D (q, a . t)	q 已被释放，不作返回值
291	倒 16	$C_N = 1 + E / N + 1$	$C'_N = E /(N+1)$	
294	倒 1	式(8-18) 第二个求和公式 $\sum_{k=0...N}$	$\sum_{k=0}^{N}$	式(8-18)前后两个求和公式格式不一致
295	1	表示对称次序下	表示中根次序下	
295	9	从左到右的对称次序	从左到右的中根次序	
301	10	1962 年，两名俄国数学家	1962 年，两名前苏联数学家	
302			图 8.23 中外结点 12 和 14 应降下来与其他一致	
303	11	插入新结点就会使	插入新结点就可能使	此处未说其是从下向上数第一个平衡系数非 0 的结点
303	13	插入新结点就会使	插入新结点就可能使	
308	倒 8	结点在对称次序下	结点在中根次序下	
308	倒 1	在对称次序下	在中根次序下	
310	倒 5	a $+ b \log_2 m$	$a + b \log_2 m$	
311	图 8.32	第三层结点中的数字都未显示全	应该都是三位数	
313	1	$K_{[m/2]-1}, P_{[m/2]-1}$	$K_{\lceil m/2 \rceil -1}, P_{\lceil m/2 \rceil -1}$	
313	1	$P_{[m/2]}, K_{[m/2]+1}, P_{[m/2]+1}$	$P_{\lceil m/2 \rceil}, K_{\lceil m/2 \rceil +1}, P_{\lceil m/2 \rceil +1}$	
313	10	代替关键词 K_1	代替关键词 K_i	
313	图 8.35	(a)中上数第二个结点中的 P_1^2	P_1^1	
313	图 8.35	(b)中上数第二个结点中的 K_1^2, P_1^2	K_1^1, P_1^1	

315	倒 7	最普通的 30 个单词之一	最普通的 31 个单词之一	
316	倒 3	因此(5)列第 9 行	因此(5)列第 10 行	
316	倒 2	第 5 行(即 E 行)	第 6 行(即 E 行)	
317	1	第(8)列 8 行	第(8)列 9 行	
317	1	来自(1)列的第 23 行	来自(1)列的第 24 行	
317	倒 5	图 8.39 给出了对应图 8.38 中表格的 30 叉树。在这个树中，	图 8.39 给出了对应图 8.38 中表格的 30 叉树，图 8.40 为对应的检索森林。在该森林中，	
318		图 8.40	最下面缺了 A, AND, ARE 三个单词	
319	2	检索表格对应的树，它是具有图 8.41 所示形式的森林。	检索表格对应的森林，其中的树具有图 8.41 所示形式。	
321		图 8.43	D 结点下面缺了一个“⊔”结点	
321	2	查找算法完全等同于 8.1 节的算法 T	查找算法完全等同于 8.3 节的算法 T	
321	4	算法 D (*ROOT*, *K* . *P*)	算法 D (*ROOT*, *K* . *p*)	
321	5	它以数字“0 < 1”为序	它以二进制数 0 < 1 为序	
321	倒 2	（NODE (*q*) 作为 NODE (*p*) 的右孩子）	（NODE (*q*) 作为 NODE (*p*) 的右孩子），$p \leftarrow q$.	
321	倒 1	算法 **D** 对于 M（$M \geq 2$）	算法 **D** 对于 M（M >2）	另外，该行文字应该为正文大字向前提
323	倒 7	mod*M* 的剩余系	mod *M* 的剩余系	中间加个空格
328	表 8.3	二进制 18 位，八进制 3 位	二进制 18 位，八进制 6 位	
332	3	$S(\lambda) = 1 + (N-1)/2M \approx 1 + 0.5\lambda$	$S(\lambda) = 1 + (N-1)/(2M) \approx 1 + 0.5\lambda$	
337	15	算法 R (TABLE[ ], *M*. *i*)	算法 R (TABLE[ ], *M*, *i* . TABLE[ ])	