《数据结构：抽象建模、实现与应用》

勘误表（2021.08.11）

第 2 章 线性表

1. 伪代码中宏常量 PARA_ERROR，DELE_FAIL，LIST_EMPTY 等可在编程实现时自行定义。

2. P14 第一行“第 1 步：GetElem(Lb，i，&bi)”修改改为“第 1 步：GetElem（Lb，i，bi）”

3. P16 算法 2.2 中 L.pData = (ElemType *)malloc(LISTINITSIZE*sizeof(ElemType); 修改为 L.pData = (ElemType *)malloc(LISTINITSIZE*sizeof(ElemType));

4. P18 算法 2.7 中 L.elem 修改为 L.pData

5. P21 算法 2.10 中 while (L.head.next != NULL) 修改为 while (L.head->next != NULL)

6. P23 算法 2.13 中 本页最后一行 L.pTail 修改为 L.tail

7. P25 算法 2.15 最后一行注释应该把 After 改为 Before，即 //DeleteElemBeforeCurNode

8. P30 算法 2.18 中需在 1.2 部分 p = pre->next; 语句后面增加一段代码，以跳过头结点：

```
if ( p == L.head )
{
    pre = p;
    p = p->next;
 }
```

第 3 章 栈和队列

1. P39 算法 3.10 中
   (1) case ‘]’: 语句多了一个缩进，需前移跟其他 case 语句对齐。
   (2) 3 处 if (StackEmpty(S)) tag = RET_ERROR_EXTRA; 代码均修改为
   if (StackEmpty(S)) { tag = RET_ERROR_EXTRA; break; }

2. P41 算法 3.11 中
   (1) 函数名中 Status CalculateExpression (char *str, double *result) 修改为
   Status CalculateExpression (char *str, double &result)
   (2) i=0; 的下面一行 GetTop(OPTR, theta1) 少 1 个分号，修改为 GetTop(OPTR, theta1);

3. P50 算法 3.16 中 Q.pBase[rear]=e; 修改为 Q.pBase[Q.rear]=e;

4. P50 算法 3.17 中 e=Q.pBase[front]; 修改为 e=Q.pBase[Q.front];

5. P51 算法 3.19 中 s->data = e; 后面增加一行代码： s->next = NULL;

6. P51 算法 3.20 中

(1) 注释部分 //1.将栈顶数据元素赋值给 e 修改为 //1.将队头数据元素赋值给 e

(2) 在注释 //1.将队头数据元素赋值给 e 下面的两行代码需修改缩进，前移，与上下行对齐。

(3) 注释部分 //Pop 修改为 //DeQueue

第 5 章 树和二叉树

1. P75 第 11 行中 构造树二叉 T 修改为 构造二叉树 T

2. P76 第 16、20 行中的 3 个变量 p 都修改为 cur_p

3. P85 算法 5.10 中

(1) 倒数第 14 行的 } 需增加向右缩进与上一个 { 对齐。

(2) 倒数第 10、11 行的 if 语句和 return 语句 需增加向右缩进，保持对齐。

4. P92 图 5.16 b）中第 8 行中的数字 2 和 3 对调顺序，修改为 3 和 2

第 6 章 图

1. P114 公式 6.2 中 vl(z) = vl(x) 修改为 vl(z) = ve(z)

2. P118 算法 6.14 中

(1) 变量定义 int flag[MAX_V] 修改为 bool flag[MAX_V]

(2) 本页倒数第 3 行注释中 flag 标志为 0 修改为 flag 标志为 false

第 7 章 查找

1. P128 7.3.2.2 节

(1) 第 7 行中 如果要查找位于 2、5、8 或 10 号位置 修改为 如果要查找位于 2、5、8 或 11 号位置

(2) 第 12 行中公式 $\lfloor log2n \rfloor + 1$ 修改为 $\lfloor log_2 n \rfloor + 1$

2. P145 第 9 行公式 $\lceil\ /2\rceil - 1$ 修改为 $\lceil m/2 \rceil - 1$

3. P152 算法 7.9 中 if (H->elem[p].key==key) 修改为

if (H->elem[p]!=NULL && H->elem[p].key==key)

4. P154 表 7.7 中，关键字 7、32 对应的比较次数 2 修改为 3。

相应的，$ASL_{平方探测法} = \frac{1}{9} \times (2+2+5+1+2+1+1+2+1) = 1.89$ 修改为

$ASL_{平方探测法} = \frac{1}{9} \times (2+2+5+1+3+1+1+3+1) = 2.11$

第 8 章 排序

1. P167 图 8.5 中 i=5 时，48 与 91 交换 修改为 91 与 66 交换。

2. P172 算法 8.10 第 2 行注释中 L.r{low].key 修改为 L.r[low].key

3. P174 算法 8.14 修改为

```
void MergeSort( SqList &L)
{    // 对顺序表 L 进行 2 路归并排序
     SqList T;
     MSort(L.r, T.r, 1, L.length);
} // MergeSort
```

4. P174 算法 8.15 中 由于 SqList 中是静态定义的数组 r, 无需动态申请空间，删除语句：T.r = (int*) malloc((high-low+1)*sizeof(RecordType));

5. P175 算法 8.16 本页第 4 行语句 Merge(L, i, i+len-1, i+2*len)-1; 修改为
   Merge(L, i, i+len-1, i+2*len-1);

6. P177 图 8.11 b) 中，数据 035、335 应放到 F[3] 和 E[3] 列。

7. P178 算法 8.18 本页第 6 行注释中“分别对各位、十位、百位处理”修改为“分别对个位、十位、百位处理”

8. P178 基数排序算法分析中的空间复杂度分析，“空间复杂度分析：由于所有记录的“分配”都存储在链式队列中，所以$S(n) = O(n)$”修改为“空间复杂度分析：由于所有记录的“分配”都存储在链式队列中，并且每次用到了r个队头队尾指针，所以$S(n) = O(n+2r)$”

9. P179 表 8.1 中， 快速排序的空间复杂度 $O(nlog_2 n)$ 修改为 $O(log_2 n)$
   基数排序的空间复杂度 $O(rd)$ 修改为 $O(n+2r)$