数据结构实用教程

习题参考解答

徐孝凯 朱上俭 编

清华大学出版社
http://www.tup.tsinghua.edu.cn

00007393

数据结构实用教程
习题参考解答

TS86/34

徐孝凯　朱上俭　编

北京航空航天大学
★ 藏 书 ★
图书馆

清华大学出版社

C0486403

(京)新登字 158 号

内 容 简 介

本书是与主教材《数据结构实用教程》一书相配套的辅助教材，它给出了主教材中大部分习题的参考解答，并对较难的习题进行了详细分析。书的最后增加了两道综合题，详细介绍了索引有序文件和散列文件进行插入、删除和查找的具体算法，它是对该课程所学知识的综合运用。

版权所有，翻印必究。
本书封面贴有清华大学出版社激光防伪标签，无防伪标签者不得销售。

书　　名：数据结构实用教程习题参考解答
作　　者：徐孝凯　朱上俭　编
出 版 者：清华大学出版社（北京清华大学校内，邮政编码：100084）
　　http://www.tup.tsinghua.edu.cn
印 刷 者：清华大学印刷厂
发 行 者：新华书店总店北京科技发行所
开　　本：787 × 1092 1/16　印张：7　字数：139 千字
版　　次：1999 年 12 月第 1 版　1999 年 12 月第 1 次印刷
书　　号：ISBN 7-302-02072-8/TP·2200
印　　数：0001 ~ 5000
定　　价：10.00 元

前　言

本书是与清华大学出版社出版的《数据结构实用教程》一书相配套的辅助教材，它给出了主教材中每一章的大部分习题的参考解答，并在最后增加了两道综合练习题。读者可以按照每道习题的要求，独立地分析问题和解决问题，并编写出相应的算法，而不要事先翻阅答案。书中所给的参考解答只有当你解题后或确实不会做时进行参考才是有益的。

数据结构是一门实践性很强的课程。对于自己编写的每一个算法，不仅要从设计思路上分析其正确性和有效性，更重要的是上机验证，并且在反复调试的过程中，通过典型的数据输入使得算法中的每条语句都被执行过，或者说不存在没有被执行过的语句或语句块。若调试过程中发现语法或逻辑错误，则要及时修改。所谓逻辑错误是指算法设计上隐含的错误，虽然算法能够被正确地编辑和连接，但运行后得不到正确的结果。通过上机运行程序不仅能验证算法的正确性，而且能加深对所学知识的理解和掌握，进而获得书本上学不到的知识。

解决一个算法问题通常要经过以下几步：①根据题目要求分析出设计思路；②根据设计思路画出相应的流程图；③根据流程图用一种计算机语言（如 C++）编写出详细的算法；④编写出能够调用该算法的完整程序；⑤上机调试和运行该程序，若发现错误则回到上述某一步再开始向下修改，通过反复调试和修改，直到获得满意的结果为止。当然对于一些简单问题，上述步骤有的可以省略，有的则可以合并。

对于要解决的同一个问题，由于所采用的数据结构可能不同，所选择的运算方法（即算法）可能不同，则编写出的程序就可能不同，但只要你的程序正确并且有效（即具有较好的时间复杂度和空间复杂度）即可。例如要对 n 个数据进行排序，既可选择数组结构，也可选择二叉排序树结构。对于数组结构，可以从快速、堆、合并、直接插入、直接选择等多种排序运算方法中任选一种，当采用不同的数据结构和排序方法时编写出的排序程序也不会相同，但它们都是正确的，通过进行时间复杂度和空间复杂度的分析可以比较出哪一种或几种更为有效。因此，不要求读者编写出的算法程序与本书所给的解答完全一致，也许你编写出的算法更优秀，综合指标更好。

书中所有算法和程序都在 Visual C++ 6.0 版本上调试通过，但由于编写时间仓促，错误和不足之处在所难免，敬请广大读者批评指正。

徐孝凯　朱上俭

1999 年 10 月

目　录

此为试读，需要完整PDF请访问：www.ertongbook.com

第一章 绪 论

1.1 有下列几种用二元组表示的数据结构，试画出它们分别对应的结构示意图（当出现多个关系时，对每个关系画出相应的结构图），并指出它们分别属于何种结构。

1. A = (K,R) 其中

   $K = \{a_1, a_2, a_3, \ldots, a_n\}$

   $R = \{\}$

2. B = (K,R)，其中

   $K = \{a,b,c,d,e,f,g,h\}$

   $R = \{r\}$

   $r = \{<a,b>, <b,c>, <c,d>, <d,e>, <e,f>, <f,g>, <g,h>\}$

3. C = (K,R)，其中

   $K = \{a,b,c,d,e,f,g,h\}$

   $R = \{r\}$

   $r = \{<d,b>, <d,g>, <b,a>, <b,c>, <g,e>, <g,h>, <e,f>\}$

4. D = (K,R)，其中

   $K = \{1,2,3,4,5,6\}$

   $R = \{r\}$

   $r = \{(1,2),(2,3),(2,4),(3,4),(3,5),(3,6),(4,5),(4,6)\}$

5. E = (K,R)，其中

   $K = \{48,25,64,57,82,36,75,43\}$

   $R = \{r_1, r_2, r_3\}$

   $r_1 = \{<48,25>, <25,64>, <64,57>, <57,82>, <82,36>, <36,75>, <75,43>\}$

   $r_2 = \{<48,25>, <48,64>, <64,57>, <64,82>, <25,36>, <82,75>, <36,43>\}$

   $r_3 = \{<25,36>, <36,43>, <43,48>, <48,57>, <57,64>, <64,75>, <75,82>\}$

解：略

1.2 设计二次多项式 $ax^2 + bx + c$ 的一种抽象数据类型，假定起名为 QUAdratic，该类型的数据部分为三个系数项 a、b 和 c，操作部分为：

1. 初始化数据成员 a、b 和 c（假定用记录类型 Quadratic 定义数据成员），每个数据成员

的默认值为0。

Quadratic InitQuadratic(float aa = 0, float bb = 0, float cc = 0);

2. 做两个多项式加法,即使对应的系数相加,并返回相加结果。

Quadratic Add(Quadratic q1, Quadratic q2);

3. 根据给定 x 的值计算多项式的值。

float Eval(Quadratic q, float x);

4. 计算方程 $ax^2 + bx + c = 0$ 的两个实数根,对于有实根、无实根和不是二次方程(即 a == 0)这三种情况都要返回不同的整数值,以便调用函数做不同的处理。

int Root(Quadratic q, float& r1, float& r2);

5. 按照 ax ** 2 + bx + c 的格式(x^2 用 x ** 2 表示)输出二次多项式,在输出时要注意去掉系数为0的项,并且当 b 和 c 的值为负时,其前不能出现加号。

void Print(Quadratic q);

请写出上面每一个操作的具体实现。

解:参考算法分别如下。

1.
```
Quadratic InitQuadratic(float aa, float bb, float cc)
{
    Quadratic q;
    q.a = aa;
    q.b = bb;
    q.c = cc;
    return q;
}
```

2.
```
Quadratic Add(Quadratic q1, Quadratic q2)
{
    Quadratic q;
    q.a = q1.a + q2.a;
    q.b = q1.b + q2.b;
    q.c = q1.c + q2.c;
    return q;
}
```

3.
```
float Eval(Quadratic q, float x)
{
    return (q.a * x * x + q.b * x + q.c);
}
```

4.
```
int Root(Quadratic q, float& r1, float& r2)
{
    if(q.a == 0) return -1;
    float x = q.b * q.b - 4 * q.a * q.c;
    if(x >= 0){
        r1 = (float)(-q.b + sqrt(x))/(2 * q.a);
```

```
            r2 = (float)( - q.b - sqrt(x))/(2 * q.a);
            return 1;
        }
        else
            return 0;
    }
```

5.
```
    void Print(Quadratic q)
    {
        if (q.a) cout << q.a << "x * * 2";
        if(q.b)
            if(q.b > 0)
               cout << " + " << q.b << "x";
            else
                cout << q.b << "x";
        if(q.c)
            if(q.c > 0)
                cout << " + " << q.c;
            else
                cout << q.c;
        cout << endl;
    }
```

1.3 用 C++ 函数描述下列每一种算法,并分别求出它们的时间复杂度。

1. 比较同一简单类型的两个数据 $x1$ 和 $x2$ 的大小,对于 $x1 > x2$、$x1 == x2$ 和 $x1 < x2$ 这三种不同情况应分别返回'>'、'='和'<'字符。假定简单类型用 SimpleType 表示,它可通过 typedef 语句定义为任一简单类型。

2. 将一个字符串中的所有字符按相反的次序重新放置。

3. 求一维 double 型数组 a[n]中的所有元素之乘积。

4. 计算 $\sum_{i=0}^{n} \frac{x^i}{i+1}$ 的值。

5. 假定一维整型数组 a[n]中的每个元素值均在[0,200]区间内,分别统计出落在[0,20),[20,50),[50,80),[80,130),[130,200]等各区间内的元素个数。

6. 从二维整型数组 a[m][n]中查找出最大元素所在的行、列下标。

解: 参考答案分别如下。

1. 时间复杂度为 $O(1)$

```
char Compare(SimpleType x1, SimpleType x2)
{
    if(x1 > x2) return '>';
    else if(x1 == x2) return '=';
    else return '<';
}
```

2. 时间复杂度为 $O(n)$

```
void Reverse(char * p)
{
    int n = strlen(p);
    for(int i = 0;i < n/2;i ++){
        char ch;
        ch = p[i];
        p[i] = p[n - i - 1];
        p[n - i - 1] = ch;
    }
}
```

3. 时间复杂度为 $O(n)$

```
double Product(double a[], int n)
{
    double p = 1;
    for(int i = 0; i < n; i ++)
        p * = a[i];
    return p;
}
```

4. 时间复杂度为 $O(n)$

```
double Accumulate(double x,int n)
{
    double p = 1,s = 1;
    for(int i = 1; i <= n; i ++){
        p * = x;
        s += p/(i + 1);
    }
    return s;
}
```

5. 时间复杂度为 $O(n)$

```
int Count(int a[], int n, int c[5])
        //用数组 c[5]保存统计结果
{
    int d[5] = {20,50,80,130,201};  //用来保存各统计区间的上限
    int i,j;
    for(i = 0;i < 5;i ++) c[i] = 0;  //给数组 c[5]中的每个元素赋初值 0
    for(i = 0;i < n;i ++)
    {
        if(a[i] < 0 || a[i] > 200)
            return 0;  //返回数值 0 表示数组中数据有错,统计失败
```

```
            for(j = 0;j < 5;j ++ )  //查找 a[i]所在的区间
                if(a[i] < d[j]) break;
            c[j] ++ ;  //使统计相应区间的元素增 1
        }
        return 1;  //返回数值 1 表示统计成功
    }
```

6. 时间复杂度为 $O(n^2)$

```
void find(int a[M][N], int m, int n, int& Lin, int& Col)
          //M 和 N 为全局常量，应满足 M>=n 和 N>=n 的条件，Lin 和 Col
          //为引用形参，它是对应实参的别名，其值由实参带回
{
    Lin = 0;Col = 0;
    for(int i = 0;i < m;i ++ )
        for(int j = 0;j < n;j ++ )
            if(a[i][j] > a[Lin][Col]) {Lin = i;Col = j;}
}
```

1.4 指出下列各算法的功能并求出其时间复杂度。

1.
```
int Prime(int n)
{
    int i = 2;
    int x = (int)sqrt(n);
    while(i <= x){
        if(n % i == 0) break;
        i ++ ;
    }
    if(i > x)
       return 1;
    else
        return 0;
}
```

2.
```
int sum1(int n)
{
    int p = 1, s = 0;
    for(int i = 1; i <= n; i ++ ){
        p *= i;
        s += p;
    }
    return s;
    }
```

3.
```
int sum2(int n)
{
    int s = 0;
    for(int i = 1; i <= n; i ++){
        int p = 1;
        for(int j = 1; j <= i; j ++)
            p * = j;
        s += p;
    }
    return s;
}
```

4.
```
int fun(int n)
{
    int i = 1,s = 1;
    while(s < n)
        s += ++ i;
    return i;
}
```

5.
```
void UseFile(ifstream& inp, int c[10])
        //假定 inp 所对应的文件中保存有 n 个整数
{
    for(int i = 0; i < 10; i ++)
        c[i] = 0;
    int x;
    while(inp >> x){
        i = x % 10;
        c[i] ++;
    }
}
```

6.
```
void mtable(int n)
{
    for(int i = 1; i <= n; i ++){
        for(int j = i; j <= n; j ++)
            cout << i << "*" << j << "="
                 << setw(2) << i * j << " ";
        cout << endl;
    }
}
```

7.
```
void cmatrix(int a[M][N], int d)
        //M 和 N 为全局整型常量
{
    for(int i = 0; i < M; i ++)
```


此为试读,需要完整PDF请访问: www.ertongbook.com

```
        for(int j = 0; j < N; j ++)
            a[i][j] * = d;
    }
```

```
8. void matrimult(int a[M][N], int b[N][L], int c[M][L])
        //M,N 和 L 均为全局整型常量
    {
        int i,j,k;
        for(i = 0;i < M;i ++)
            for(j = 0;j < L;j ++)
                c[i][j] = 0;
        for(i = 0;i < M;i ++)
            for(j = 0;j < L;j ++)
                for(k = 0;k < N;k ++)
                    c[i][j] += a[i][k] * b[k][j];
    }
```

解：参考答案分别如下。

1. 判断 n 是否是一个素数，若是则返回数值 1，否则返回 0。该算法的时间复杂度为 $O(\sqrt{n})$。

2. 计算 $\sum_{i=1}^{n} i!$ 的值。时间复杂度为 $O(n)$。

3. 计算 $\sum_{i=1}^{n} i!$ 的值。时间复杂度为 $O(n^2)$。

4. 求出满足不等式 $1+2+3+\cdots+i \geqslant n$ 的最小 i 值。时间复杂度为 $O(\sqrt{n})$。

5. 利用数组 c[10]中的每个元素 c[i]对应统计出 inp 所联系的整数文件中个位值同为 i 的整数个数。时间复杂度为 $O(n)$。

6. 打印出一个具有 n 行的乘法表，第 i 行($1 \leqslant i \leqslant n$)中有 $n-i+1$ 个乘法项，每个乘法项为 i 与 j($i \leqslant j \leqslant n$)的乘积。时间复杂度为 $O(n^2)$。

7. 使数组 a[M][N]中的每一个元素均乘以 d 的值。时间复杂度为 $O(M \times N)$。

8. 矩阵相乘，即 a[M][N] × b[N][L]→c[M][L]。时间复杂度为 $O(M \times N \times L)$。

1.5 设计集合的一种抽象数据类型。

集合是由若干个同一类型元素组成的、元素之间不存在任何关系的一种数据结构。通常，一个集合用一对花括号括起来，元素之间用逗号分隔。一个集合中的元素来自于一个数据集，并且不允许出现重复的元素。如对于 1～n 之间的整数集，它共包含有 2^n 个不同的集合，其中{}表示空集，{1,2,...,n}表示全集。假定一个整数集为 1～3，则在它之上可以构成的 8(2^3)个集合为：

{}, {1}, {2}, {3}, {1,2}, {1,3}, {2,3}, {1,2,3}

在 C++ 语言中，可用一个整型数组来表示一个集合，若一个数组元素的值为 0，则表示相应元素不在集合中，若为 1 则表示相应元素存在于集合中。如对于整数集 1～6 之上的一个集合{1,4,5}，则用整型数组 a[7]表示(a[0]元素未用)为：

0	1	2	3	4	5	6
	1	0	0	1	1	0

对于集合运算通常有并(∪)、交(∩)和属于(∈)等。两个集合的并的结果仍为一个集合,它包含有两个集合中的所有元素,当然不允许出现重复。两个集合的交的结果也仍为一个集合,其中的每一个元素同时属于两个集合。属于运算是判断一个元素是否存在于一个集合之中,若存在则返回真(True),否则返回假(False)。假定 x = {1,4,5},y = {2,4},则 x∪y = {1,2,4,5},x∩y = {4},1∈x 为真,2∈x 为假。

假定在 1 ~ SETSIZE 整数集(SETSIZE 为一个整型全局常量)上建立集合,抽象数据类型名用 SET 表示,该类型的数据部分为一个整型数组 m[SETSIZE + 1],用于保存一个集合,操作部分为:

1. 对一个集合中的所有元素清 0,假定用记录类型 Set 定义数据成员,即 Set 类型的定义为:struct Set {int m[SETSIZE + 1]}。该操作就是对 Set 类型的一个对象初始化,使其数组 m 域中的每个元素被置为 0。

   void InitSet(Set& s);

2. 利用整型数组 a[n]初始化数据成员,即置一个集合中的 m[a[i]]为 1(0≤i≤n - 1),如假定 a[3] = {1,3,6},则相应集合中的 m[1]、m[3]和 m[6]元素应被置为 1。

   void InitSet(Set& s, int a[], int n);

3. 重载加法运算符实现两个集合的并运算。

   Set operator + (Set s1, Set s2);

4. 重载乘法运算符实现两个集合的交运算。

   Set operator * (Set s1, Set s2);

5. 重载按位异或(^)运算符实现属于(∈)运算。

   Boolean operator ^(int elt, Set s);

6. 向一个集合中加入一个元素。

   void Insert(Set& s, int n);

7. 从一个集合中删除一个元素。

   void Delete(Set& s, int n);

8. 重载流插入操作符(<<),用于输出一个集合。

   ostream& operator << (ostream& ostr, Set& s);

请写出上述每一个操作的具体实现,并上机调试,检查其正确性。

假定使用该抽象数据类型的主函数如下:

```
#include<iostream.h>
const int SETSIZE=50;
struct Set{
    int m[SETSIZE+1];
};
enum Boolean{False, True};
void main()
```

```
{
    Set x, y, z;
    int a[5] = {2,5,10,20,50};
    InitSet(x);
    Insert(x,8);
    Insert(x,a[3]);
    InitSet(y,a,5);
    cout << x << y << endl;
    z = x + y;cout << z;
    z = x * y;cout << z;
    if(10^y)
        cout << "True" << endl;
    else
        cout << "False" << endl;
    Delete(y,20);
    cout << y;
}
```

则运行结果为：

```
{8,20,}
{2,5,10,20,50,}

{2,5,8,10,20,50,}
{20,}
True
{2,5,10,50,}
```

解：参考答案分别如下。

1.
```
void InitSet(Set& s)
{
    for(int i = 1; i <= SETSIZE; i ++ )
        s.m[i] = 0;
}
```

2.
```
void InitSet(Set& s, int a[], int n)
{
    for(int i = 0; i < n; i ++ )
        s.m[a[i]] = 1;
}
```

3.
```
Set operator + (Set s1, Set s2)
{
    Set s;
    InitSet(s);
    for(int i = 1; i <= SETSIZE; i ++ )
```

```
        if((s1.m[i] == 1)||(s2.m[i] == 1))
            s.m[i] = 1;
    return s;
}
```

4.
```
Set operator * (Set s1, Set s2)
{
    Set s;
    InitSet(s);
    for(int i = 1; i <= SETSIZE; i ++)
        if((s1.m[i] == 1)&&(s2.m[i] == 1))
            s.m[i] = 1;
    return s;
}
```

5.
```
Boolean operator ^(int elt, Set s)
{
    if(s.m[elt] == 1)
        return True;
    else
        return False;
}
```

6.
```
void Insert(Set& s, int n)
{
    s.m[n] = 1;
}
```

7.
```
void Delete(Set& s, int n)
{
    s.m[n] = 0;
}
```

8.
```
ostream& operator << (ostream& ostr, Set& s)
{
    ostr << '{';
    for(int i = 1; i <= SETSIZE; i ++)
        if(s.m[i] == 1)
            ostr << i << ',';
    ostr << '}' << endl;
    return ostr;
}
```

第二章　线性表

2.1　在下面的每个程序段中，假定线性表 La 的类型为 List，元素类型 ElemType 为 int，并假定每个程序段是连续执行的，试写出每个程序段执行后所得到的线性表 La。

1.
```
InitList(La);
int a[] = {48,26,57,34,62,79};
for(i = 0; i < 6; i ++)
    InsertFront(La,a[i]);
TraverseList(La);
```

2.
```
InitList(La);
for(i = 0; i < 6; i ++)
    Insert(La,a[i]);
TraverseList(La);
```

3.
```
Insert(La,56);
DeleteFront(La);
InsertRear(La, DeleteFront(La));
TraverseList(La);
```

4.
```
for(i = 1; i <= 3; i ++) {
    int x = GetElem(La,i);
    if(x % 2 == 0) Delete(La,x);
}
TraverseList(La);
```

5.
```
ClearList(La);
for(i = 0; i < 6; i ++)
    InsertRear(La,a[i]);
Delete(La, a[5]);
Sort(La);
Insert(La,a[5]/2);
TraverseList(La);
```

解：参考答案分别如下：

1. (79, 62, 34, 57, 26, 48)
2. (26, 34, 48, 57, 62, 79)
3. (48, 56, 57, 62, 79, 34)
4. (56, 57, 79, 34)
5. (26, 34, 39, 48, 57, 62)

2.2 对于第一题的前四个程序段，假定 La 的类型为构造单链表的数组类型 AlinkList，元素类型 ElemType 仍为 int，并假定每个程序段是连续执行的，试画出每个程序段执行后所得到的单链表的示意图，要求在示意图的每个指针上注明具体数值。

解：

为了排版方便，假定采用以下输出格式表示单链表的示意图：每个括号内的数据表示一个元素结点，其中第一个数据为元素值，第二个数据为后继结点的指针，第一个元素结点前的数值为表头指针。参考答案分别如下。

1. (7(79,6), (62,5), (34,4), (57,3), (26,2), (48,0))
2. (3(26,5), (34,2), (48,4), (57,6), (62,7), (79,0))
3. (2(48,8), (56,4), (57,6), (62,7), (79,5), (34,0))
4. (8(56,4), (57,7), (79,5), (34,0))

2.3 对于 List 类型的线性表，编写出下列每个算法。

1. 从线性表中删除具有最小值的元素并由函数返回，空出的位置由最后一个元素填补，若线性表为空则显示出错信息并退出运行。

解：

```
ElemType DMValue(List& L)
    //从线性表中删除具有最小值的元素并由函数返回，空出的位置
    //由最后一个元素填补，若线性表为空则显示出错信息并退出运行
{
    if(ListEmpty(L)) {
        cerr<<"List is Empty!"<<endl;
        exit(1);
    }
    ElemType x;
    x=L.list[0];
    int k=0;
    for(int i=1; i<L.size; i++) {
        ElemType y=L.list[i];
        if (y<x) {x=y; k=i; }
    }
    L.list[k]=L.list[L.size-1];
    L.size--;
    return x;
}
```

2. 从线性表中删除第 i 个元素并由函数返回。

解：

```
int Delete1(List& L, int i)
```


此为试读，需要完整PDF请访问：www.ertongbook.com