第 1 章　绪论

教材中练习题及参考答案

1. 简述数据与数据元素的关系与区别。

答：凡是能被计算机存储、加工的对象统称为数据，数据是一个集合。数据元素是数据的基本单位，是数据的个体。数据元素与数据之间的关系是元素与集合之间的关系。

2. 采用二元组表示的数据逻辑结构 S=<*D*，*R*>，其中 *D*={*a*，*b*，…，*i*}，*R*={*r*}，*r*={<*a*，*b*>，<*a*，*c*>，<*c*，*d*>，<*c*，*f*>，<*f*，*h*>，<*d*，*e*>，<*f*，*g*>，<*h*，*i*>}，问关系 *r* 是什么类型的逻辑结构？哪些结点是开始结点，哪些结点是终端结点？

答：该逻辑结构为树形结构，其中 *a* 结点没有前驱结点，它是开始结点，*b*、*e*、*i* 和 *g*、结点没有后继结点，它们都是终端结点。

3. 简述数据逻辑结构与存储结构的关系。

答：在数据结构中，逻辑结构与计算机无关，存储结构是数据元素之间的逻辑关系在计算机中的表示。存储结构不仅将逻辑结构中所有数据元素存储到计算机内存中，而且还要在内存中存储各数据元素间的逻辑关系。通常情况下，一种逻辑结构可以有多种存储结构，例如，线性结构可以采用顺序存储结构或链式存储结构表示。

4. 简述数据结构中运算描述和运算实现的异同。

答：运算描述是指逻辑结构施加的操作，而运算实现是指一个完成该运算功能的算法。它们的相同点是，运算描述和运算实现都能完成对数据的“处理”或某种特定的操作。不同点是，运算描述只是描述处理功能，不包括处理步骤和方法，而运算实现的核心则是设计处理步骤。

5. 数据结构和数据类型有什么区别？

答：数据结构是相互之间存在一种或多种特定关系的数据元素的集合，一般包括三个方面的内容，即数据的逻辑结构、存储结构和数据的运算。而数据类型是一个值的集合和定义在这个值集上的一组运算的总称，如C语言中的short int数据类型是由-32768～32767（16位机）的整数和+、-、*、/、%等运算符构成。

6. 在 C/C++中提供了引用运算符，简述其在算法描述中的主要作用。

答：在算法设计中，一个算法通常用一个或多个 C/C++函数来实现，在 C/C++函数之间传递参数时有两种情况，一是从实参到形参的单向值传递，二是实参和形参之间的双向值传递。对形参使用引用运算符，即在形参名前加上“&”，不仅可以实现实参和形参之间的双向值传递，而且使算法设计简单明晰。

7. 有以下用 C/C++语言描述的算法，说明其功能：

```
void fun(double &y,double x,int n)
{   y=x;
    while (n>1)
    {   y=y*x;
        n--;
    }
}
```

答：本算法的功能是计算 $y=x^n$。

8. 用 C/C++语言描述下列算法，并给出算法的时间复杂度。

（1）求一个 n 阶整数数组的所有元素之和。

（2）对于输入的任意 3 个整数，将它们按从小到大的顺序输出。

（3）对于输入的任意 n 个整数，输出其中的最大和最小元素。

答：（1）算法如下：

```
int sum(int A[N][N],int n)
{   int i,j,s=0;
    for (i=0;i<n;i++)
        for (j=0;j<n;j++)
            s=s+A[i][j];
    return(s);
}
```

本算法的时间复杂度为 $O(n^2)$。

（2）算法如下：

```
void order(int a,int b,int c)
{   if (a>b)
    {   if (b>c)
            printf("%d,%d,%d\n",c,b,a);
        else if (a>c)
            printf("%d,%d,%d\n",b,c,a);
        else
            printf("%d,%d,%d\n",b,a,c);
    }
    else
    {   if (b>c)
        {   if (a>c)
                printf("%d,%d,%d\n",c,a,b);
            else
                printf("%d,%d,%d\n",a,c,b);
        }
        else printf("%d,%d,%d\n",a,b,c);
    }
}
```

本算法的时间复杂度为 $O(1)$。

（3）算法如下：

```
void maxmin(int A[],int n,int &max,int &min)
```

```
{    int i;
     min=min=A[0];
     for (i=1;i<n;i++)
     {    if (A[i]>max)  max=A[i];
          if (A[i]<min)  min=A[i];
     }
}
```

本算法的时间复杂度为 O(n)。

9. 设 3 个表示算法频度的函数 f、g 和 h 分别为：

$f(n)=100n^3+n^2+1000$

$g(n)=25n^3+5000n^2$

$h(n)=n^{1.5}+5000n\log_2 n$

求它们对应的时间复杂度。

答：$f(n)=100n^3+n^2+1000=O(n^3)$，$g(n)=25n^3+5000n^2=O(n^3)$

当$n\to\infty$时，$\sqrt{n}>\log_2 n$，所以$h(n)=n^{1.5}+5000n\log_2 n=O(n^{1.5})$。

10. 分析下面程序段中循环语句的执行次数。

```
int j=0, s=0, n=100;
do
{    j=j+1;
     s=s+10*j;
} while (j<n && s<n);
```

答：j=0，第 1 次循环：j=1，s=10。第 2 次循环：j=2，s=30。第 3 次循环：j=3，s=60。第 4 次循环：j=4，s=100。while 条件不再满足。所以，其中循环语句的执行次数为 4。

11. 设 n 为正整数，给出下列 3 个算法关于问题规模 n 的时间复杂度。

（1）算法 1

```
void fun1(int n)
{    i=1, k=100;
     while (i<=n)
     {    k=k+1;
          i+=2;
     }
}
```

（2）算法 2

```
void fun2(int b[], int n)
{    int i, j, k, x;
     for (i=0;i<n-1;i++)
     {    k=i;
          for (j=i+1;j<n;j++)
               if (b[k]>b[j]) k=j;
          x=b[i];b[i]=b[k];b[k]=x;
     }
}
```

（3）算法 3

```
void fun3(int n)
{   int i=0,s=0;
    while (s<=n)
    {   i++;
        s=s+i;
    }
}
```

答：（1）设 while 循环语句执行次数为 $T(n)$，则：

$i=2T(n)+1\leqslant n$，即 $T(n)\leqslant(n-1)/2=O(n)$。

（2）算法中的基本运算语句是 if ($b[k]>b[j]$) $k=j$，其执行次数 $T(n)$为：

$$T(n)=\sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1}1=\sum_{i=0}^{n-2}(n-i-1)=\frac{n(n-1)}{2}=O(n^2)$$

（3）设 while 循环语句执行次数为 $T(n)$，则：

$s=1+2+\cdots+T(n)=\dfrac{T(n)(T(n)+1)}{2}\leqslant n$，则 $T(n)=O(\sqrt{n})$。

12. 有以下递归算法用于对数组 $a[i..j]$的元素进行归并排序：

```
void mergesort(int a[],int i,int j)
{   int m;
    if (i!=j)
    {   m=(i+j)/2;
        mergesort(a,i,m);
        mergesort(a,m+1,j);
        merge(a,i,j,m);
    }
}
```

求执行 mergesort(a，0，$n-1$)的时间复杂度。其中，merge(a，i，j，m)用于两个有序子序列 $a[i..m]$和 $a[m+1..j]$的合并，是非递归函数，它的时间复杂度为 O(合并的元素个数)。

答：设 mergesort(a，0，$n-1$)的执行时间为 $T(n)$，分析得到以下递归关系：

$T(n)=O(1)$ $\quad n=1$

$T(n)=2T(n/2)+O(n)$ $\quad n>1$

其中，O(n)为 merge()所需的时间，设为 cn（c 为常量）。因此：

$$T(n)=2T(\frac{n}{2})+cn=2(2T(\frac{n}{2^2})+\frac{cn}{2})+cn=2^2T(\frac{n}{2^2})+2cn=2^3T(\frac{n}{2^3})+3cn$$

$$\vdots$$

$$=2^kT(\frac{n}{2^k})+kcn=2^kO(1)+kcn$$

由于 $\dfrac{n}{2^k}$趋近于 1，则 $k=\log_2 n$。所以 $T(n)=2^{\log_2 n}O(1)+cn\log_2 n=n+cn\log_2 n=O(n\log_2 n)$。

13. 描述一个集合的抽象数据类型 ASet，其中所有元素为正整数，集合的基本运算包括：

（1）由整数数组 $a[0..n-1]$创建一个集合。

（2）输出一个集合的所有元素。

（3）判断一个元素是否在一个集合中。

（4）求两个集合的并集。

（5）求两个集合的差集。

（6）求两个集合的交集。

在此基础上设计集合的顺序存储结构，并实现各基本运算的算法。

答：抽象数据类型 ASet 的描述如下：

```
ADT ASet
{    数据对象：D={ di | 0≤i≤n，n为一个正整数}
     数据关系：无。
     基本运算：
          createset( &s，a，n)：创建一个集合s;
          dispset( s)：输出集合s;
          inset(s,e)：判断元素e是否在集合s中。
          void add(s1，s2，s3)：s3=s1∪s2;          //求集合的并集
          void sub(s1，s2，s3)：s3=s1-s2;          //求集合的差集
          void intersection(s1，s2，s3)：s3=s1∩s2;  //求集合的交集
}
```

设计集合的顺序存储结构类型如下：

```
typedef struct                    //集合结构体类型
{    int data[MaxSize];           //存放集合中的元素，其中 MaxSize 为常量
     int length;                  //存放集合中实际元素个数
} Set;                            //将集合结构体类型用一个新类型名 Set 表示
```

采用 Set 类型的变量存储一个集合。对应的基本运算算法设计如下：

```
void createset(Set &s,int a[],int n) //创建一个集合
{    int i;
     for (i=0;i<n;i++)
          s.data[i]=a[i];
     s.length=n;
}
void dispset(Set s)          //输出一个集合
{    int i;
     for (i=0;i<s.length;i++)
          printf("%d ",s.data[i]);
     printf("\n");
}
bool inset(Set s,int e)          //判断 e 是否在集合 s 中
{    int i;
     for (i=0;i<s.length;i++)
          if (s.data[i]==e)
               return true;
```

```
    return false;
}
void add(Set s1,Set s2,Set &s3) //求集合的并集
{   int i;
    for (i=0;i<s1.length;i++)   //将集合 s1 的所有元素复制到 s3 中
        s3.data[i]=s1.data[i];
    s3.length=s1.length;
    for (i=0;i<s2.length;i++)   //将 s2 中不在 s1 中出现的元素复制到 s3 中
        if (!inset(s1,s2.data[i]))
        {   s3.data[s3.length]=s2.data[i];
            s3.length++;
        }
}
void sub(Set s1,Set s2,Set &s3) //求集合的差集
{   int i;
    s3.length=0;
    for (i=0;i<s1.length;i++)   //将 s1 中不出现在 s2 中的元素复制到 s3 中
        if (!inset(s2,s1.data[i]))
        {   s3.data[s3.length]=s1.data[i];
            s3.length++;
        }
}
void intersection(Set s1,Set s2,Set &s3) //求集合的交集
{   int i;
    s3.length=0;
    for (i=0;i<s1.length;i++)       //将 s1 中出现在 s2 中的元素复制到 s3 中
        if (inset(s2,s1.data[i]))
        {   s3.data[s3.length]=s1.data[i];
            s3.length++;
        }
}
```